CAROUSEL BANNERS

Carousel banners (also referred to as carousel sets) enable marketers to drive conversion by easily creating interactive rotating promotional content, and delivering it to any screen.

Creating and modifying content featured in promotional banners can be time-consuming, limiting your ability to quickly publish new content, or make it more targeted.

Carousel Banners enable you to quickly create or modify rotating banners, add interactivity such as hotspots that link to product detail and related resources, and deliver them to any screen—letting you bring new promotional content to market faster.

1. NAVIGATE TO CAROUSEL SAMPLE

A. Navigate to Assets → Files → Summit [1]

B. Click on the asset card titled “Sports” to preview the existing set

C. Explore this sample by changing slides, click on the hotspots and image maps

---

[1] *http://localhost:4502/assets.html/content/dam/summit*

2. EDIT THE CAROUSEL SET

We'll explore some features by modifying this carousel set, and adding some new content.

A. Click "Edit" in the top left of the screen to open the Carousel Banner Editor

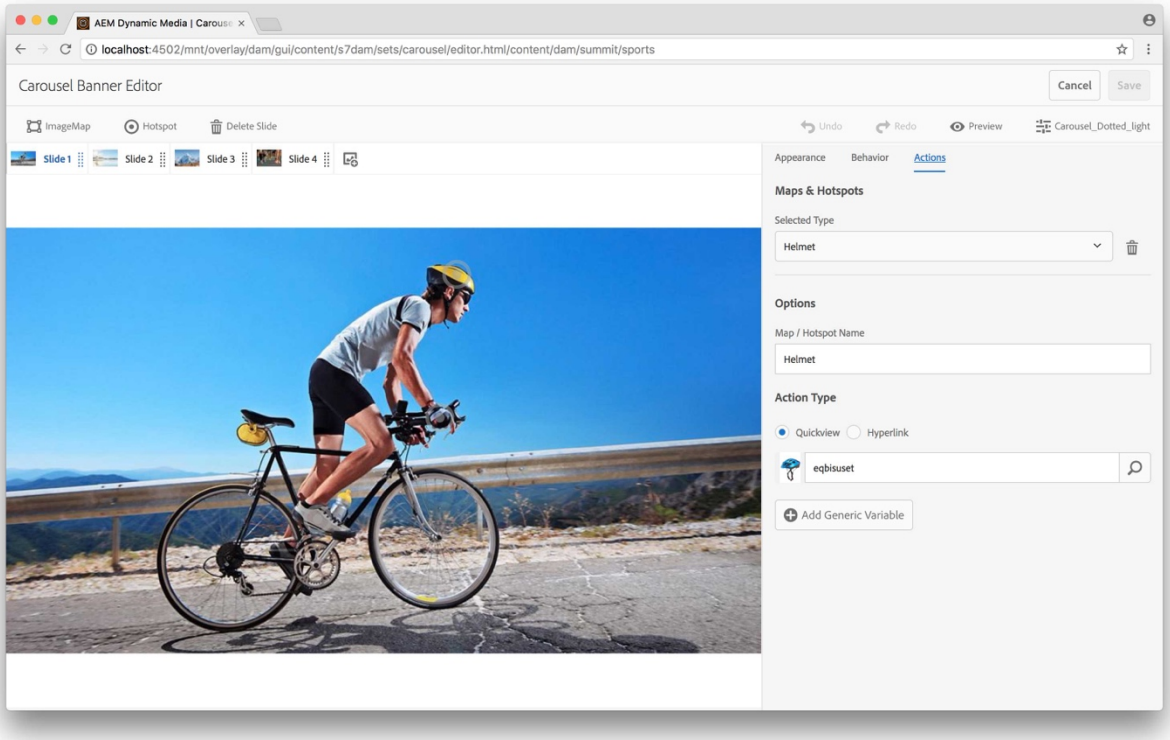


Figure 1: Carousel Banner Editor

3. THE SLIDE BAR

A. Click the various slides to browse the existing images in this set

Figure 2: Slide Bar

B. Let's add some images by clicking the "Add Slide" button on the right of the slide bar

Figure 3: Add Slide button

C. Select some images using the asset picker

i. Navigate to Summit → Demo Assets → Activities → Running

ii. Select the top two images “Running Couple Mountain” and “Running Desert Woman” by hovering over the two thumbnails and clicking on them.

Figure 4: Selecting assets

iii. Click the blue “Select” button on the top right of the asset picker.

D. Try re-arranging the order of the slides by dragging the slide bar.

4. HOTSPOTS AND IMAGEMAPS

Carousel sets support both hotspots and image maps allowing marketers to link product detail to regions of interest.

A. Observe the regions of interest defined on the various slides

Figure 5: Hotspot target

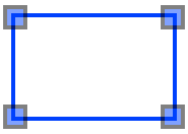


Figure 6: Image map region

B. Click on a hotspot or image map to see the name and configured action in the “Actions” panel

The action type can be a quickview referencing a product in the system, or an external hyperlink.

C. Let’s add a hotspot. First select the “Running Couple Mountain” slide that we added earlier.

D. Click on the “Hotspot” button located in the action bar above the slide bar

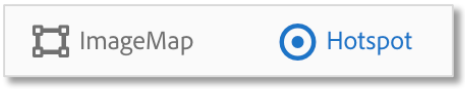


Figure 7: Select “Hotspot” in action bar

E. Place the hotspot by clicking on the purple gloves within the slide (pictured in figure 8)

F. Configure an action from the “Actions” panel on the right

i. Type a descriptive name for the hotspot, such as “purple gloves”

ii. Let’s associate this hotspot with an existing product by selecting the “Quickview” action type, and then clicking on the search icon in the field below to open the product picker.

Figure 8: Placing hotspot for gloves

iii. Select the “gloves” from the product picker.

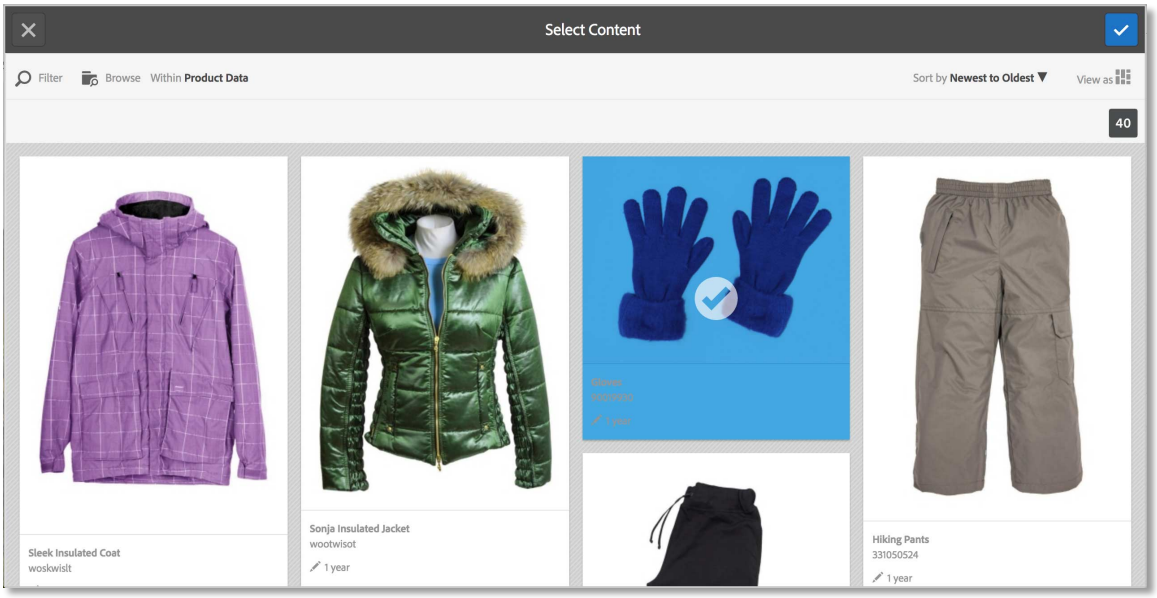


Figure 9: Product picker

G. Place another hotspot for the running pants

i. Click the “Hotspot” action bar button again to place a hotspot for the running pants

ii. In the “Actions” panel, type a name for the new hotspot, such as “running pants”

iii. Open the product picker, select “Faba Running Pants”.

H. Switch to the “Running Desert Woman” slide, and toggle the image map control

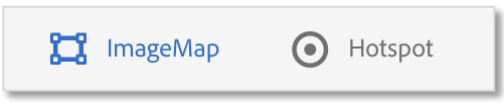


Figure 10: Select “ImageMap” in action bar

I. Drag an image map region over the blue jacket

Figure 11: Image map region

J. Configure a Hyperlink action

i. Type a name for the image map, such as “blue jacket”
ii. Associate this image map with a hyperlink by selecting “Hyperlink” as the action type
iii. Enter a test URL of your choice.

K. Click the “Preview” button to see what we have so far.

5. CUSTOMIZING APPEARANCE

The carousel editor enables live configuration and preview of appearance customizations

A. Select the “Appearance” tab to customize the hotspot appearance

i. Select the "ImageMapEffect" component and expand the "Background" widget to select custom art, or simply use one of the out of the box gallery icons.
ii. Click the gallery button (highlighted in figure 12)
iii. Select an out of the box hotspot icon

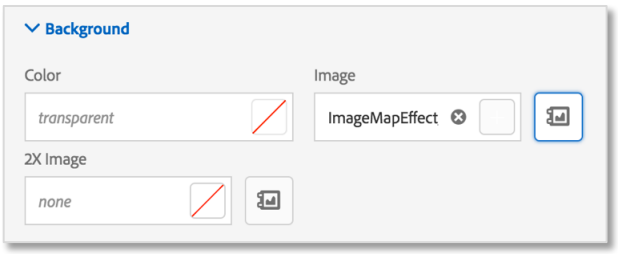


Figure 12: Background Widget

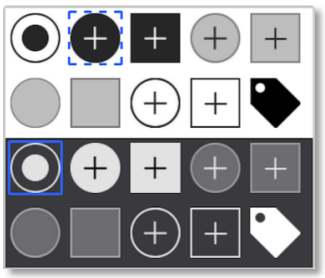


Figure 13: Hotspot gallery

B. Open the “Shadow” widget

i. Set the Shadow type to “outset” and configure the desired effect

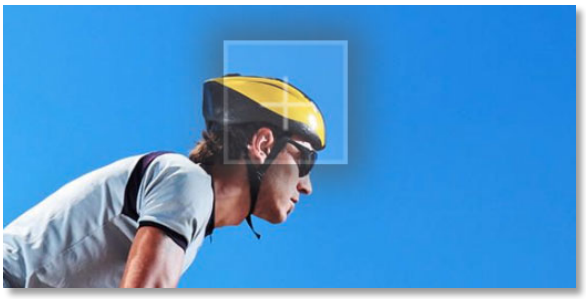


Figure 14: Live preview of hotspot

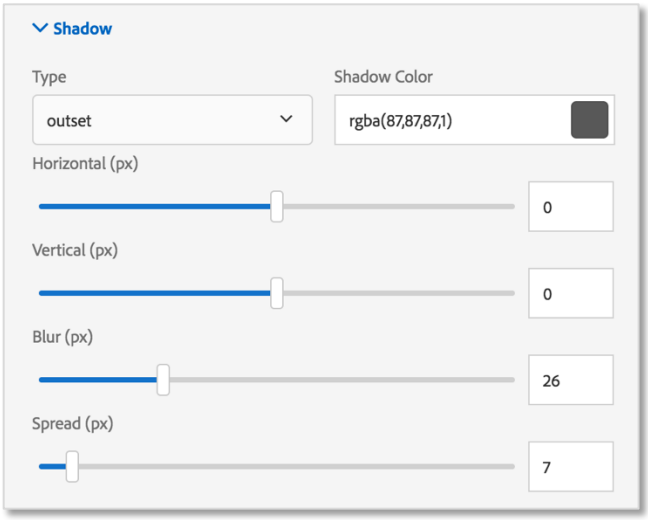


Figure 15: Shadow Widget

C. Select the “SetIndicator” component

i. Inside the position widget, adjust the bottom slider to any desired value

ii. Under the “Dotted Style” section, adjust the sliders within the size widget

iii. Explore setting a shadow or a border

6. CUSTOMIZING BEHAVIOR

In addition to "Appearance" we can adjust behavioral modifiers in the "Behavior" tab. The carousel viewer is built from several components that expose modifiers that can be customized.

A. Select the "CarouselView" component under the Behavior tab

i. Open the "Slide Transition" widget to reveal animation options.

ii. Change the default "fade" animation to "side".

B. Select the “SetIndicator” component under the Behavior tab

i. Change the “Set Indicator Mode” to “numeric” to observe the effect

7. PREVIEW THE SET AND SAVE

Once satisfied with our appearance and behavior customizations we can save the carousel set. This tutorial covers just a small sample of the many aspects that can be configured.

A. Click “Save”

B. Provide a new preset name such as “Demo”

The carousel editor makes a distinction between the carousel content and the viewer configuration made in the “Appearance” and “Behavior” tabs.

Viewer configuration is saved separately in a viewer preset. This provides a lot of flexibility since it allows many carousel sets to share common viewer appearance and behavior configuration.

Note: User custom user presets can be overwritten while out of the box presets are read-only. Customizations to out of the box presets require us to save as a new preset.

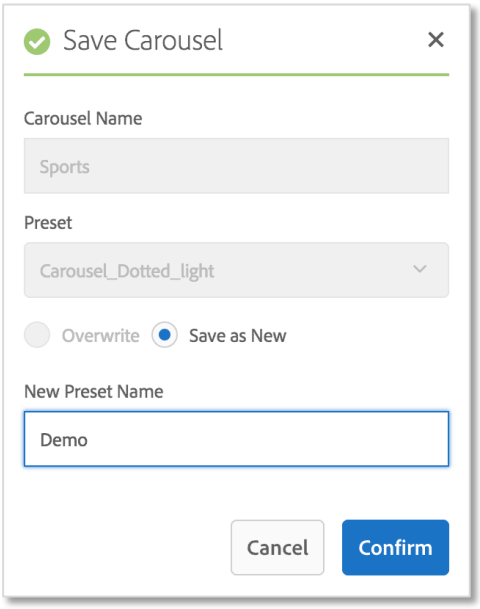
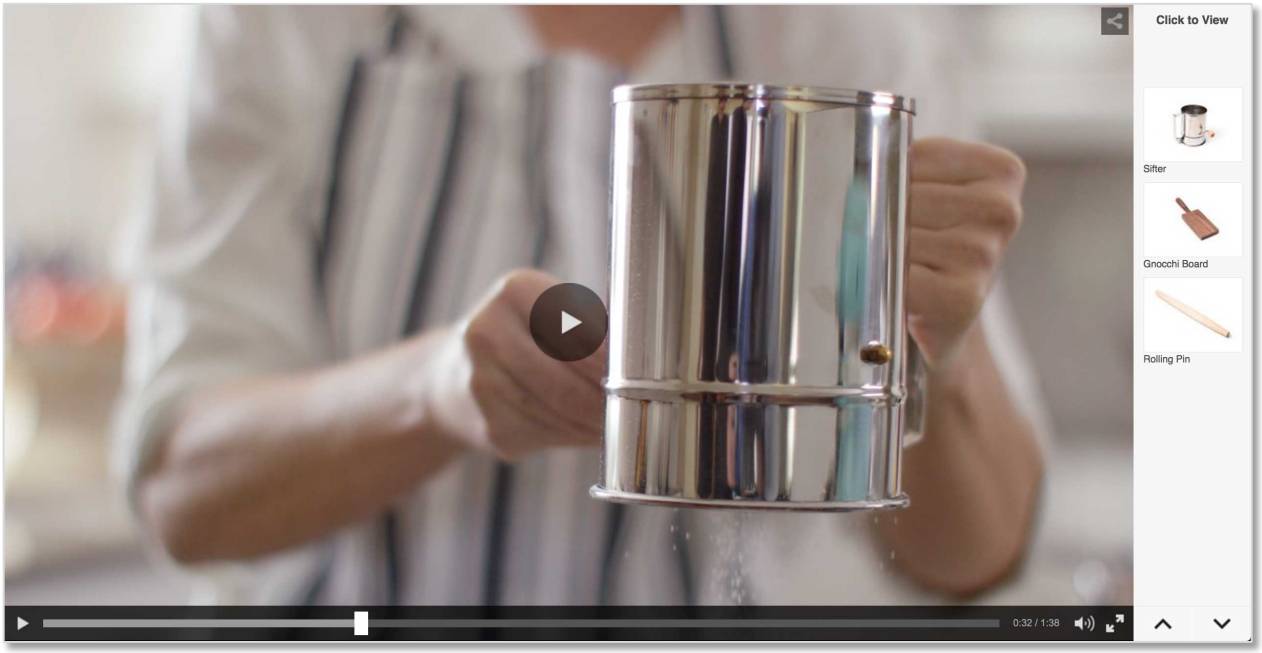


Figure 16: Save carousel dialog

C. Click the “Confirm” button to save and exit the Carousel Banner Editor.

INTERACTIVE VIDEO

Create interactive videos that drive conversion. Add product detail to defined segments of video to allow customers to link directly to their shopping cart for immediate purchase, as well as other services.



1. NAVIGATE TO INTERACTIVE VIDEO SAMPLE

A. Navigate to Assets → Files → Summit → Shoppable Video [2]

B. Click on the asset card titled “Kitchen Video” to preview the video

2. EXPLORING THE INTERACTIVE VIDEO EDITOR

We’ll explore some features by modifying an existing interactive video, and adding some new content.

A. Click “Edit” in the top left of the screen to open the Interactive Video Editor

B. First, let’s preview the existing video, click the “Preview” button in the top right corner

---

[2] *http://localhost:4502/assets.html/content/dam/summit/adobe-axis-demo*

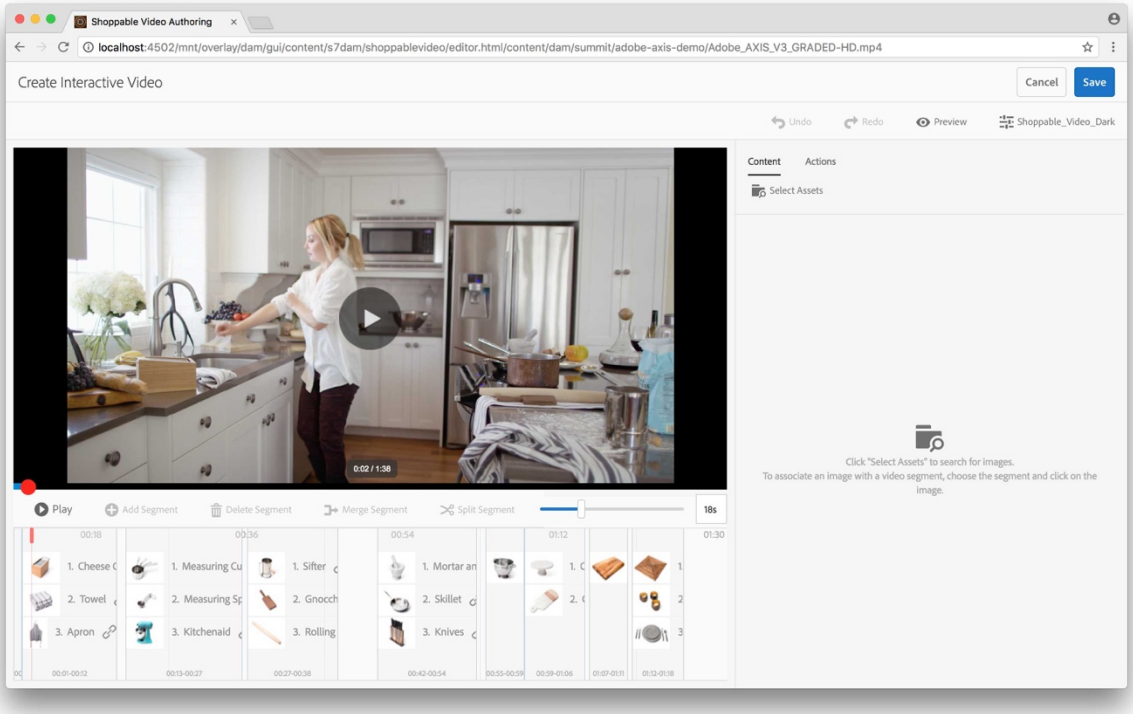


Figure 17: Interactive Video Editor

C. Adjust the “Timeline Scale Slider” to scale and examine the timeline contents more easily

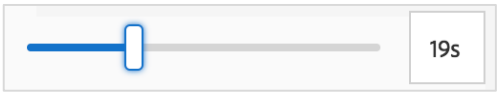


Figure 18: Timeline Scale Slider

D. Click on a segment in the timeline to adjust and inspect the associated products

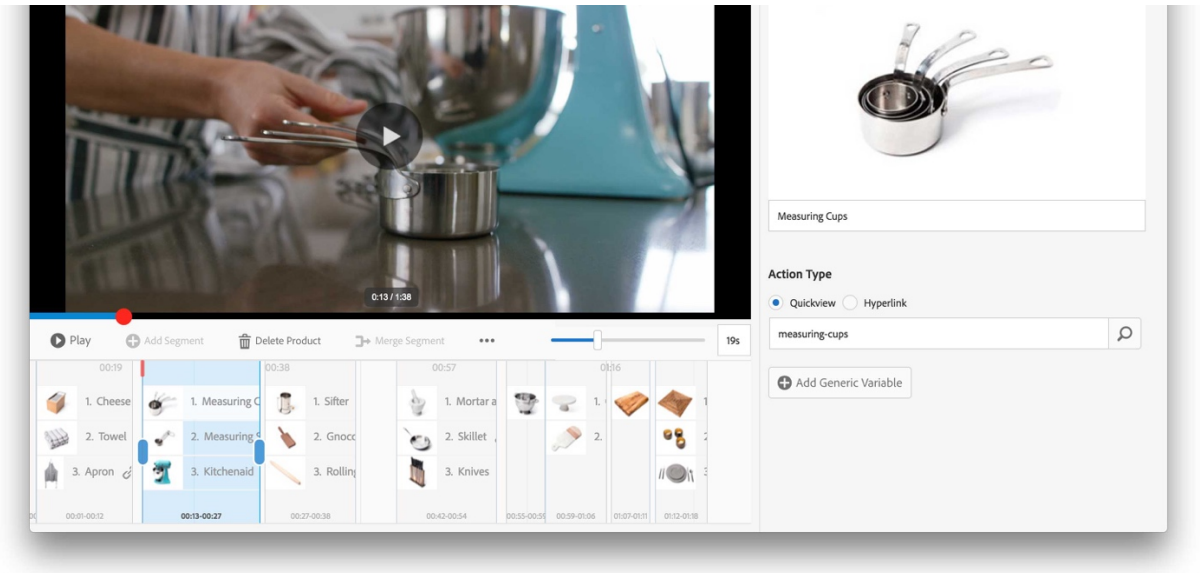


Figure 19: Select segment and product

i. Try adjusting segment duration by dragging the blue handles
ii. Clicking on a segment handle moves the playhead to that position and displays the frame of video at that position.

3. ADDING PRODUCTS TO OUR CONTENT WORKSPACE

Let's complete this incomplete interactive video by adding some missing products. First, let's add the desired products to our workspace.

A. Click the "Content" tab, then click "Select Assets" to open the asset picker.

i. Navigate to Summit → Shoppable Video → Products
ii. Select the following images by hovering over the thumbnails and clicking on them: Glass, Wine Bucket, Candle Dish, Candles, Wine Decanter, Small Glass

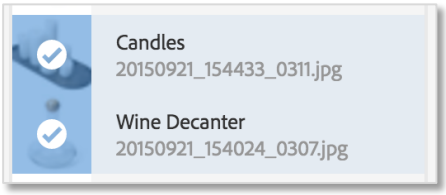


Figure 20: Selecting multiple products for interactive video

iii. Click the blue "Select" button on the top right of the asset picker.

4. SEGMENTING VIDEO

To start segmentation of the video, we must choose a position and click "Add Segment".

A. First seek to position 1:18

i. To do this, drag the red video seek thumb to quickly jump around the timeline

B. Click the "Add Segment" button

i. Let's adjust this new segment so that it spans from about 1:18 to 1:23 on the timeline. This will represent the wine decanter and glass.

C. Associate segment with products

i. Make sure only the single segment is highlighted by clicking on it in the timeline
ii. Scroll through the "Content" tab and click on Wine Decanter and Glass to add them to the selected segment.
iii. Click on the respective products in the segment to customize their name and action type. This is similar to the Carousel Banner example. In this case, we can leave the defaults as is.

D. Seek to position 1:24

E. Click "Add Segment" to add a final segment

i. Adjust the new segment to span from 1:24 to 1:35

F. Associate segment with products

   i. Follow the same steps from part C above, however this time click on the following products: Candles, Candle Dish, Wine Bucket, Small Glass.

5. PREVIEW AND SAVE SHOPPABLE VIDEO

A. Click on the “Preview” button to see the completed video

B. Click on the “Shoppable_Video_Dark” drop down select list to choose the “Shoppable_Video_Light” viewer preset.

   Viewer presets configure the appearance and behavior of the viewer itself. To configure the appearance and behavior of the interactive video viewer we can accomplish this using the Viewer Preset Editor. [3]

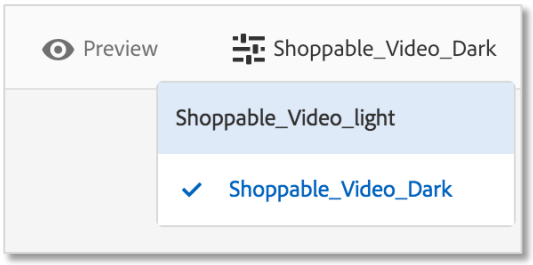


Figure 21: Preview button and preset selector

C. Once satisfied with the interactive video, we can click the blue “Save” button to save changes and exit the editor.

[3] *http://localhost:4502/mnt/overlay/dam/gui/content/s7dam/viewerpresets/viewerpresets.html*

DYNAMIC MEDIA VIEWERS SDK

The Dynamic Media Viewers SDK (also known as the Scene7 HTML5 Viewers SDK) provides a set of JavaScript-based components for custom viewer development. The viewers are web-based applications that allow for rich media content served by AEM and Adobe Scene7 to be embedded in web pages.

The components are designed to work on Android, Apple iOS devices, and desktop browsers including Internet Explorer 9 and later. This enables you to provide a single workflow for all supported platforms.

Figure 22: A custom zoom viewer that allows users to browse various images in a set

1. LOCATE BASIC ZOOM VIEWER EXAMPLE

Our goal will be to build the viewer in figure 22 above. We will begin by reviewing and augmenting a basic existing example.

A. Navigate browser to example in CRXDE

i. In Chrome (on the lab computer), open the bookmark “CRXDE – basicviewer.jsp”

ii. If bookmark is unavailable, manually navigate to CRXDE: http://localhost:4502/crx/de/

- *Inside CRXDE, make sure to navigate to node: /apps/summit-viewer/components/basicviewer/basicviewer.jsp*[4]

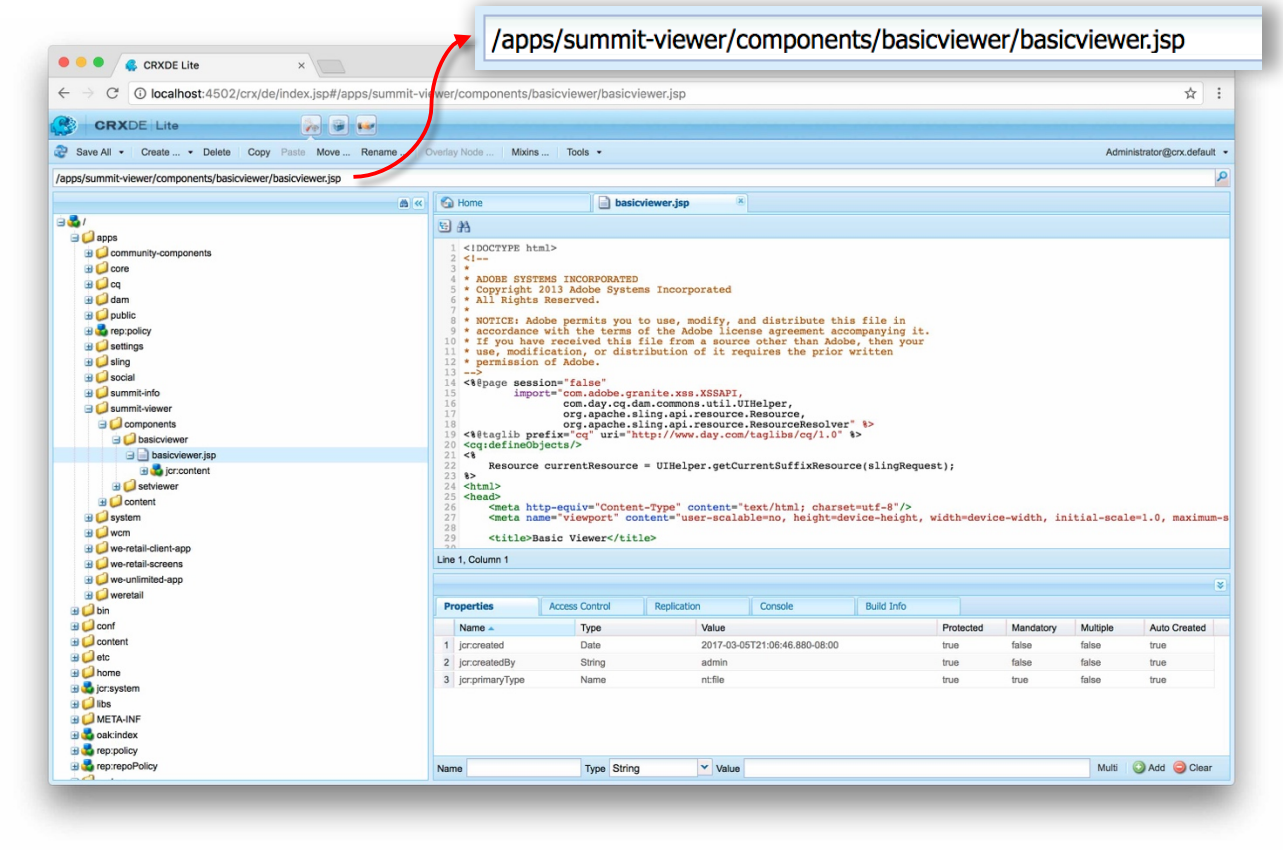


Figure 23: Custom viewer code for basicviewer.jsp in CRXDE

B. Run/open the custom viewer in a separate tab for preview purposes during development

i. In Chrome (on the lab computer), open the bookmark “summit-viewer” in a new tab

ii. If bookmark is unavailable, manually navigate to: http://localhost:4502/summit-viewer.html/content/dam/summit/climbing

iii. Try out the basic zoom viewer, double click the main viewer image to zoom in and out

2. CREATING A CUSTOM VIEWER COMPONENT

The Dynamic Media Viewers SDK was designed to stand on its own. It works both inside and outside AEM environments. We have created a simple component to wrap and integrate our custom viewer so that it naturally handles AEM asset resources from the URL.

[4] *http://localhost:4502/crx/de/index.jsp#/apps/summit-viewer/components/basicviewer/basicviewer.jsp*

A. Node structure

To work gracefully within AEM we have modeled a standard component and content node structure.

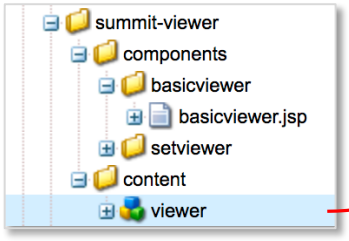


Figure 24: Node structure

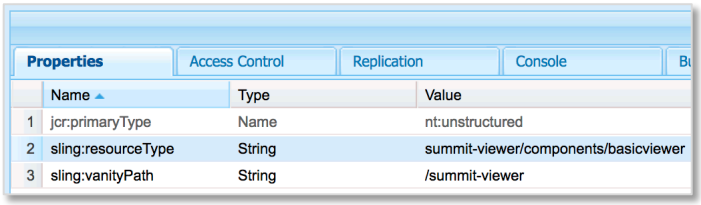


Figure 25: Properties on "viewer" content node

The content node “viewer” has “sling:resourceType” set to load scripts within the “basicviewer” component folder.

In addition, “sling:vanityPath” has been set to “/summit-viewer”. This allows us to access the viewer using a simplified URL.

B. Let’s open and analyze the contents of “basicviewer.jsp” in CRXDE

Our custom viewer component is a simple script that generates an entire HTML document. The JSP reads the resource provided in the URL and passes it to the SDK. This can be seen on line 22 and line 90. The remainder of the JSP outputs the static HTML and JavaScript code as is for our custom viewer.

```
<%
    Resource currentResource = UIHelper.getCurrentSuffixResource(slingRequest);
%>
```

Figure 26: Code to read resource from URL

3. THE STRUCTURE OF A VIEWER

A. Include the core SDK script “Utils.js” inside the head element:

```
    <!--
        Include Utils.js before you use any of the SDK components. This file contains
        SDK utilities and global functions that are used to initialize the viewer and
        load the viewer components. The path to Utils.js specifies which version of the
        SDK that the viewer uses. Here we can use a relative path since both our custom
        viewer and the SDK are served from within the same AEM instance.
    -->
    <script language="javascript" type="text/javascript"
            src="/etc/dam/viewers/s7sdk/3.2/js/s7sdk/utils/Utils.js"></script>
```

Figure 27: Including Utils.js

B. Import the necessary component modules for a basic zoom viewer:

```
<!--
    Add an "include" call to download the module of each component that is needed
    for the viewer. Check the API documentation to see a complete list of components
    and their modules.
-->
<script language="javascript" type="text/javascript">
    s7sdk.Util.lib.include('s7sdk.common.Container');
    s7sdk.Util.lib.include('s7sdk.image.ZoomView');
</script>
```

Figure 28: Include related component modules

C. The core initialization logic of a viewer application

```
<script language="javascript" type="text/javascript">

    /* We create a self-running anonymous function to encapsulate variable scope.
       Placing code inside such a function is optional, but this prevents variables
       from polluting the global object. */
    (function () {

        // Initialize the SDK
        s7sdk.Util.init();

        /* Create an instance of the ParameterManager component to collect viewer
           configuration. Configuration can come from a viewer preset, URL, or the
           HTML page itself. The ParameterManager component also sends an event once
           all of the needed files are loaded and the configuration parameters have
           been processed. */
        var params = new s7sdk.ParameterManager();

        /* Event handler for the s7sdk.Event.SDK_READY event that is dispatched by
           ParameterManager. Within this handler we can instantiate the various viewer
           components. All components (except the ParameterManager) must be instantiated
           within this handler. */
        function initViewer() {

            /* Default modifiers can be defined directly on the ParameterManager
               instance. Here we configure the viewer to operate using AEM resource
               paths by default. */
            params.push("aemmode", "1");
            params.push("asset", "<%= xssAPI.getValidHref(currentResource.getPath()) %>");

            // Instantiate viewer components ...
        }
        /* Register the event handler for the s7sdk.Event.SDK_READY event that is dispatched
           by the ParameterManager. This event is dispatched once modifiers have been
           processed and it is safe to initialize the viewer. */
        params.addEventListener(s7sdk.Event.SDK_READY, initViewer, false);

        /* After registering the handler for the s7sdk.Event.SDK_READY event we can invoke
           ParameterManager.init() to continue viewer initialization and configuration. */
        params.init();
    }());

</script>
```

Figure 29: Core initialization of viewer application

4. CREATING AN IMAGE SET VIEWER

Let's augment the provided "basicviewer.jsp" example to give users the ability to select images from a set. To do this we will need to add the MediaSet and Swatches components.

A. Add the following SDK includes underneath the existing ones on line 61:

```
s7sdk.Util.lib.include('s7sdk.set.MediaSet');
s7sdk.Util.lib.include('s7sdk.set.Swatches');
```

Figure 30: Include modules for the MediaSet and Swatches components

B. Update the variable list with the following:

```
var mediaSet, container, zoomView, swatches;
```

Figure 31: Add "mediaSet" and "swatches" variables

C. Instantiate the MediaSet and Swatches components inside the initViewer function. Add the following code underneath the instantiation of the ZoomView component.

Be sure to instantiate the Swatches instance after the ZoomView and Container components, otherwise the stacking order will cover and hide the Swatches.

```
// Create MediaSet component to retrieve and parse set information
mediaSet = new s7sdk.set.MediaSet(null, params, "mediaSet");

// Register an event handler to determine when the set has been parsed
mediaSet.addEventListener(s7sdk.event.AssetEvent.NOTF_SET_PARSED, onSetParsed, false);

// Create Swatches component and register an event handler for swatch selection
swatches = new s7sdk.set.Swatches("s7container", params, "mySwatches");
swatches.addEventListener(s7sdk.event.AssetEvent.SWATCH_SELECTED_EVENT,
    onSwatchSelected, false);
```

Figure 32: Instantiation of MediaSet and Swatches

D. Add the event handler for MediaSet directly underneath the code above

```
/* Event handler for the s7sdk.event.AssetEvent.NOTF_SET_PARSED event dispatched by
   MediaSet to assign the asset to the Swatches once parsing is complete. */
function onSetParsed(event) {

    // set media set for Swatches to display
    var mediasetDesc = event.s7event.asset;
    swatches.setMediaSet(mediasetDesc);

    // select the first swatch by default
    swatches.selectSwatch(0, true);
}
```

Figure 33: MediaSet event handler

E. Add the event handler for Swatches to determine when the user has made a selection

```
/* Event handler for s7sdk.event.AssetEvent.SWATCH_SELECTED_EVENT events dispatched
   by Swatches so that we can switch the image in the ZoomView when a different swatch
   is selected. */
function onSwatchSelected(event) {
   zoomView.setItem(event.s7event.asset);
}
```

Figure 34: Swatch selection handler

F. Let's have the swatches resize to take the entire viewer width. Add a call in the resizeViewer function to resize the swatches whenever the user resizes their browser.

```
swatches.resize(width, swatches.getHeight());
```

Figure 35: Resize swatches to container width

G. Finally, let's add some CSS to define the appearance of the Swatches component. Add the following within the style element near the top of the page:

```
.s7swatches {
    /* Align swatches to bottom of viewer */
    bottom: 0;
    left: 0;
    right: 0;
    height: 150px;
}
```

Figure 36: Adding CSS for Swatches component

H. Let's adjust the size of the Swatches component

i. Change the height defined for ".s7swatches" in the style element from 150px to 250px

ii. Add the following rule to make the thumbs larger. The Swatches component will automatically request the higher resolution thumbs to compensate.

```
.s7swatches .s7thumb {
    width: 150px;
    height: 150px;
}
```

Figure 37: Increase size of thumbs

5. SAVE AND PREVIEW YOUR VIEWER

A. Save basicviewer.jsp in CRXDE

B. Open the “summit-viewer” tab

i. Refresh the page to preview our enhancements

Figure 38: Final preview of our custom viewer

C. Try changing the asset resource in the URL to the “Sports” carousel

i. Change the asset in the URL from “climbing” to “sports”
http://localhost:4502/summit-viewer.html/content/dam/summit/sports
Our custom viewer is able to render the carousel set! However, the hotspot and image map regions do not show up since our custom viewer does not implement them.

RESOURCES

A. Dynamic Media (General Information)
http://www.adobe.com/marketing-cloud/enterprise-content-management/dynamic-media.html

B. Samples of Dynamic Media Viewers
https://marketing.adobe.com/resources/help/en_US/s7/vlist/vlist.html

C. Working with Dynamic Media (AEM 6.2 specific)
https://docs.adobe.com/docs/en/aem/6-2/author/assets/dynamic-media.html

D. Viewers Reference Guide
https://marketing.adobe.com/resources/help/en_US/s7/viewers_ref/

E. Introduction to the Scene7 HTML5 Viewer SDK Tutorial
https://marketing.adobe.com/resources/help/en_US/s7/viewers_ref/c_tutorial.html