Enhanced Templates Guide

Version 1.3
Updated 10/23/14

Table of Contents

Introduction

KVStudio 5 supports Enhanced Templates for KnowledgeVision presentations. Compared with the Legacy Templates that were available in KVStudio 4 and earlier, Enhanced Templates are built using standard HTML5 technologies (HTML, CSS, JavaScript) that are broadly supported across different web browsers and mobile devices. Furthermore, these templates can be designed to adapt responsively to the size and capabilities of the viewer’s screen, allowing for

optimized viewer experiences on smartphones and tablets.

Legacy Templates vs. Enhanced Tempates

Legacy Templates were, until the release of KVStudio 5, the only type of templates available in KnowledgeVision. They are rendered using Flash Player technology, which is not supported on many mobile devices, especially iPhone and iPad. As a result, the Legacy Templates render fully functional, customized, branded experiences when Flash Player is available, but they automatically switch to a limited-functionality, fixed-format layout when loaded on devices that do not support Flash Player. Although this approach provides an optimized viewer experience across devices, it does not allow content creators to control, customize, or brand that experience.

Enhanced Templates move all of the functional and design elements of a template into HTML5. The video and slide images may still be rendered with Flash Player when it is available, but all of the other components, such as chapter navigation, footnotes, attachments, playlist navigation, transcripts, and zoom controls, as well as standard elements like images, text, and links, are rendered in standard HTML5. As a result, viewers see the same overall experience regardless of what device they're using to watch your presentation. Furthermore, the design of the template can be customized using standard web technologies, particularly CSS and JavaScript. Whereas Legacy Templates are constructed with limited, proprietary tools, Enhanced Templates can be built using almost any common website design tool, or directly with HTML, CSS, and JavaScript code.

In general, Enhanced Templates support most of the same functional components that are available in Legacy Templates. One difference you may notice, however, is that the video and slide elements have been combined into a single media player component, called the ComboPlayer. The positioning, sizing, and display of the video and slide elements within the ComboPlayer is entirely controlled by properties that are set on the project in KVStudio, rather than template properties. These project properties are as follows:

- Programmed zoom levels
    - Extremes of the zoom control now show only slides or video
- Slide and video aspect ratios
    - Standard (4:3)
    - Widescreen (16:9)
- Layout options:
    - Video left
    - Video right
    - Audio and slides
    - Video only

As a result, certain designs that were supported in Legacy Templates are not possible in Enhanced Templates, such as layouts where the video and slide elements were separated or fixed in particular dimensions. However, this approach eliminates the need to create specialized

templates for each of the configurations listed above. It also provides a more consistent experience to viewers across different presentations and different viewing devices.

Another important difference between Legacy Templates and Enhanced Templates relates to sizing. Legacy Templates use fixed dimensions, which determine the width and height (in pixels) that the template will occupy on the viewer's screen. Enhanced Templates can also be assigned fixed dimensions, which may be appropriate if you plan to embed the presentation within a fixed layout on your website. However, Enhanced Templates also support flexible dimensions, allowing them to dynamically stretch to fill the space available to them. If you embed the presentation on your website, a flexible template will automatically fill the space you give it. If you instead link directly to the presentation, a flexible template will fill the entire browser window. In particular, this flexibility enables Enhanced Templates to support Responsive Design.

Responsive Design

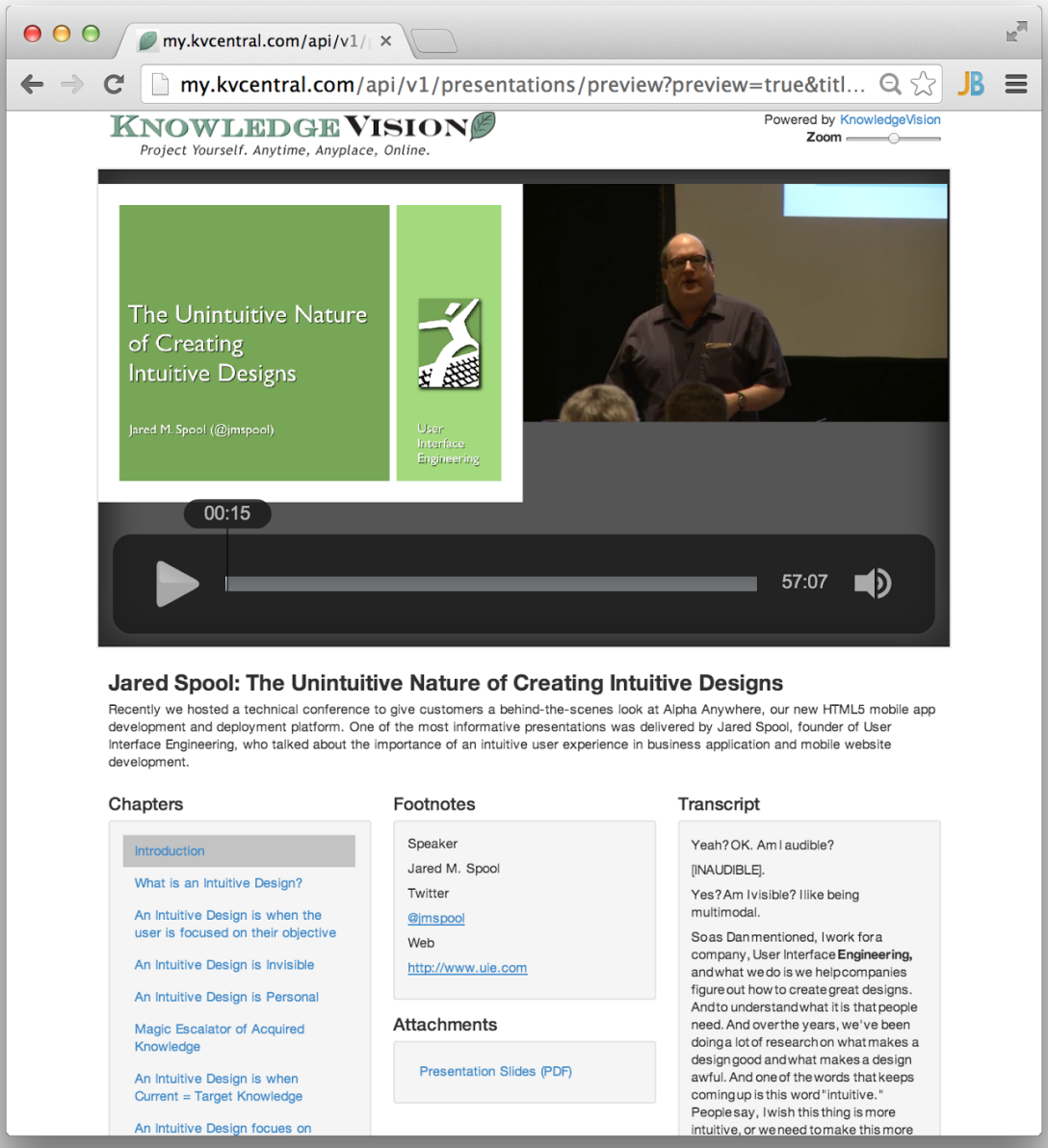
Responsive Design is a web design concept that has become popular over the last few years as trends in technology have radically shifted. Whereas web content was at one time only accessible from your desktop web browser, it is now being consumed increasingly on mobile devices, particularly smartphones and tablets. These devices offer screen sizes that are typically smaller than standard desktop displays, and most provide a touch-screen interface rather than traditional point-and-click with a mouse. As a result, the design paradigms that have worked well on the desktop rarely translate successfully to mobile devices. In fact, the differences in screen size between smartphones and tablets are usually significant enough they also require specialized designs.

One strategy that web designers have employed is to create completely separate versions of their websites that are optimized for mobile viewers. This approach is problematic because it requires maintaining two websites instead of one, which creates major technical and logistical challenges. Instead, more and more designers have moved to using Responsive Design, which relies on HTML5 to detect the screen size and capabilities of a viewer's device and then dynamically adjust the page design to provide the most optimized experience. With Responsive Design, layouts that display multiple columns on a desktop browser can compress to a single column on a smartphone. Text can be enlarged on smaller devices to make it more easily readable without forcing the viewer the zoom in. Information that was hidden when the viewer held her tablet in landscape orientation can be revealed when she rotates to the wider portrait orientation.

Because Enhanced Templates are built with HTML5, they can leverage many of the innovations of Responsive Design to deliver optimized viewer experiences across devices. Whereas a viewer on a desktop might like to see chapters, footnotes, and attachments all on one screen, a viewer on a smartphone may prefer to tab between this information, one section at a time. When a viewer rotates his tablet from portrait to landscape, a navigation menu might open automatically to reveal more options. It may be helpful to display the presentation title and

description more prominently for an on-the-go mobile viewer who just needs enough information about the video to decide to bookmark it for later. Responsive Design makes this level of on-demand optimization possible, and it ensures that every type of viewer has the best possible user experience.

Below are screenshots illustrating how the same presentation, published with the same responsive Enhanced Template, appears across three different devices:



Carrier 3:35 PM 100%

my.kvcentral.com

my.kvcentral.com/api/v1/presentations/preview?preview=true&title=20140128_Jared_Spool&accountId=11&subAccountId=-1&...

KNOWLEDGE VISION
Project Yourself. Anytime, Anyplace, Online.

Powered by KnowledgeVision
Zoom

Jared Spool: The Unintuitive Nature of Creating Intuitive Designs

Recently we hosted a technical conference to give customers a behind-the-scenes look at Alpha Anywhere, our new HTML5 mobile app development and deployment platform. One of the most informative presentations was delivered by Jared Spool, founder of User Interface Engineering, who talked about the importance of an intuitive user experience in business application and mobile website development.

Footnotes

Speaker

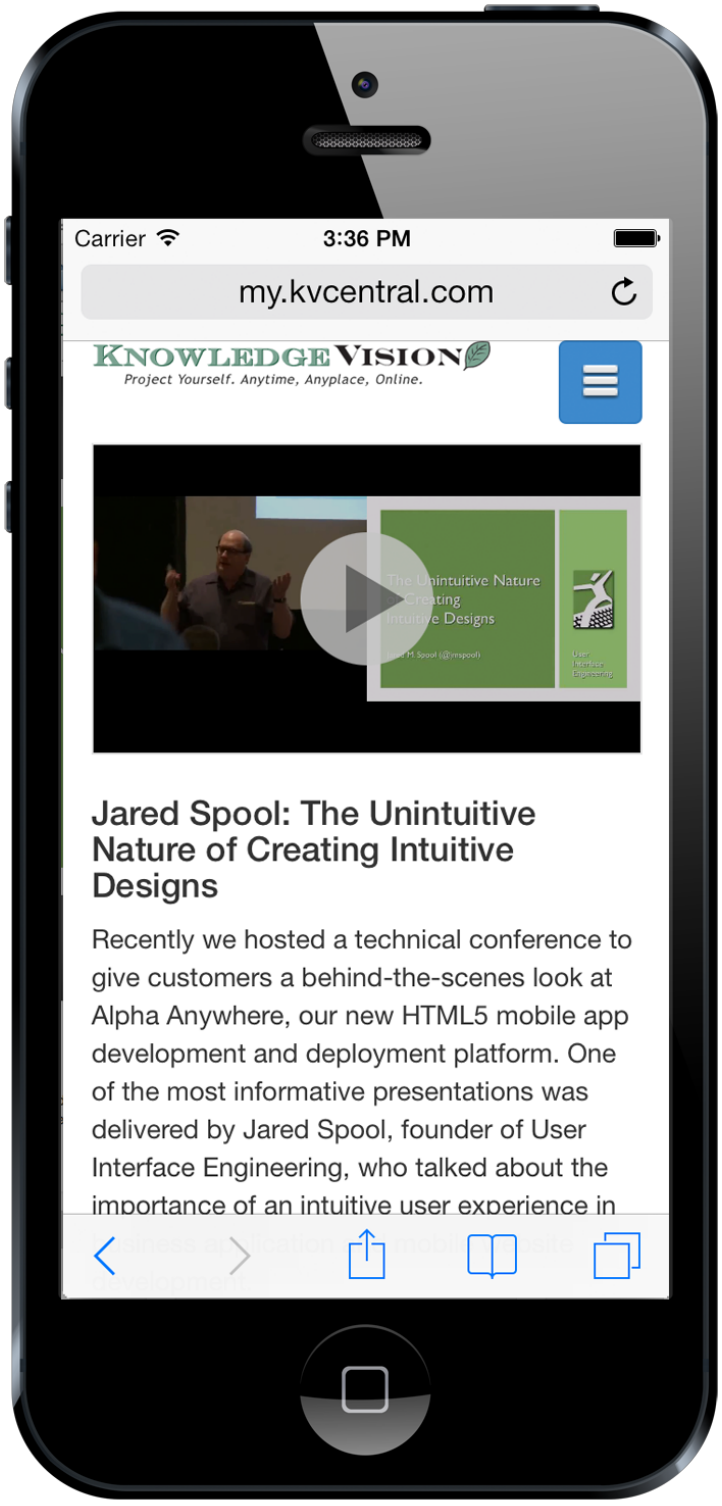
Jared M. Spool

Twitter

@jmspool

Web

http://www.uie.com



Responsive Design generally requires more design time because the designer must consider all of these different scenarios, rather than just one. Fortunately, there are several open-source toolkits that take away most of the technical complexity of implementing Reponsive Design. In

particular, the Twitter Bootstrap framework (http://getbootstrap.com/) offers a robust Responsive Design layout framework, which we use this in most of our standard templates. However, there are also situations where you may prefer to build a more consistent, but less responsive, template, because you expect most viewers to access your content from desktop browsers. As a template designer, you have the full range of options available to you.

Device Limitations

Enhanced Templates open up an incredible array of opportunities to control and customize the look, feel, and functionality of your KnowledgeVision presentations across different viewing devices. However, there are still fundamental technical limitations imposed by certain devices that may affect the user experience for your viewers. It is important to understand these limitations and design your templates appropriately.

Flash Player

Despite the advancements in web standards, Flash Player still provides certain important capabilities for video streaming and animation that are not yet supported consistently in HTML5. As a result, Enhanced Templates use Flash Player to render the video stream and slide images for a presentation when it is available. If Flash Player is not supported on the device, as is the case on iPhone, iPad, and many Android devices, the Enhanced Template will revert to using native HTML5 video and image elements. In this situation, the following limitations take effect:

- PowerPoint animation effects are not currently supported.
- Webcam recordings created in KVStudio and KVQuick are not currently supported.
- Uploaded video files that were not encoded for HTML5 delivery may not be supported.
- Live streaming video may not be supported.
- YouTube videos are not currently supported.
- If your video is being delivered by a third-party provider like Brightcove, Ooyala or Kaltura, the third-party player controls and chrome may not appear.

Inline Video

The Apple iPhone and some Android smartphones impose a technical limitation that prevents embedded HTML5 video from playing inline on a webpage. This means that when a viewer taps to play an embedded video, the browser automatically opens the video fullscreen in its native video player. The viewer can pause the video to return to the webpage where it was embedded, but it is not possible for the viewer to simultaneously play the video and view the content of the webpage.

To accommodate for this limitation, KnowledgeVision generates a special video file that composites the original video stream and slide images when your presentation is published. The Enhanced Template automatically detects if the viewer is watching on a device with this limitation, and if so, it swaps in this composited version of the video and slides.

This means that as a template designer, you cannot rely on the viewer seeing the other components in the template while they are watching the video. If your presentations include important footnote links or attachments, you may want to include a special notification that only appears for smartphone viewers, informing them to pause the video to access this content. Also, the zoom control feature will not function in this situation, so you may want to use Responsive Design to hide it from view.

Touchscreen Interface

Many mobile devices provide a sophisticated touchscreen interface to interact with applications and content. This interface can create some limitations, but it can also provide great opportunities. One of main limitations is there is no concept of “hovering” the mouse when using a touchscreen, so it is important to avoid relying on this technique for key functionality. Another consideration to keep in mind is that fingertips are much less accurate than mouse pointers, so interactive controls should be large and well-spaced to make it easy for viewers to tap on them. On the plus side, with a little advanced JavaScript programming, you may be able to take advantage of sophisticated multi-touch gestures like swiping and pinching to build an engaging interface with your presentation content.

Building Enhanced Templates

KnowledgeVision provides access to a variety of standard Enhanced Templates that offer basic customization options for things like the background color, logo image, and text labels. However, if you would like to take full advantage of all of the customization options that Enhanced Templates have to offer, you can build your template using HTML and CSS.

Template Structure

Every Enhanced Template begins with an HTML file. While a template package may optionally contain additional files such as images, CSS, and JavaScript, it must contain an HTML file that provides the structure of the template. When the template is loaded by the KVPlayer at runtime, your HTML code will be dynamically injected into the `<body>` element of the webpage.

Understanding Components

Components are the key building blocks of any Enhanced Template. Each component is a functional element that enhances the viewer experience by providing interactive navigation controls, access to additional content and resources, or helpful supplementary information. Many components, such as the ChapterNavigator, ThumbnailNavigator, and Footnotes elements, dynamically update their appearance as the video plays to provide synchronized context. Others,

including the Attachments, PresentationTitle, PresentationDescription components, remain static throughout the video playback.

As a template designer, you can leverage components to create an engaging, interactive viewer experience, without having to do any of the programming to make them actually function. The KVPlayer handles all of the complex behavior of each component, allowing you to focus your energy on the design elements of the template.

Each component has a special element tag that you can use to inject it into your template HTML. These tags are custom to KnowledgeVision templates, and will not be recognized outside the context of the KVPlayer. At runtime, the KVPlayer generates the actual, functional HTML elements in place of each component tag.

For example, the following sample code includes three components: PresentationTitle, ChapterNavigator, Footnotes:

Template Code

```
<h1><presentation-title></presentation-title></h1>
<h3>Chapters</h3>
<chapter-navigator></chapter-navigator>
<h3>Footnotes</h3>
<footnotes></footnotes>
```

At runtime, the KVPlayer replaces each component tag with generated HTML elements containing the actual information from the presentation:

Generated Code

```
<h1><div class="presentation-title">Introduction to Content
Marketing</div></h1>
<h3>Chapters</h3>
<ul class="chapter-navigator">
     <li class="chapter-navigator-item-selected">
          <a href="#">Introduction</a>
     </li>
     <li class="chapter-navigator-item">
          <a href="#">What is Content Marketing?</a>
     </li>
     <li class="chapter-navigator-item">
          <a href="#">How do I get started?</a>
     </li>
</ul>
<h3>Footnotes</h3>
<div class="footnotes">Interested in learning more? Give us a
call!</div>
```

Of course, you are not limited to only using custom KnowledgeVision components in your Enhanced Template. You can use any standard HTML elements, such as `<img>` and `<a>` tags to add images and links.

Components Guide

The following guide lists each custom component available in Enhanced Templates, along with a screenshot, description, example code, generated code, and list of attributes, if applicable.

ComboPlayer

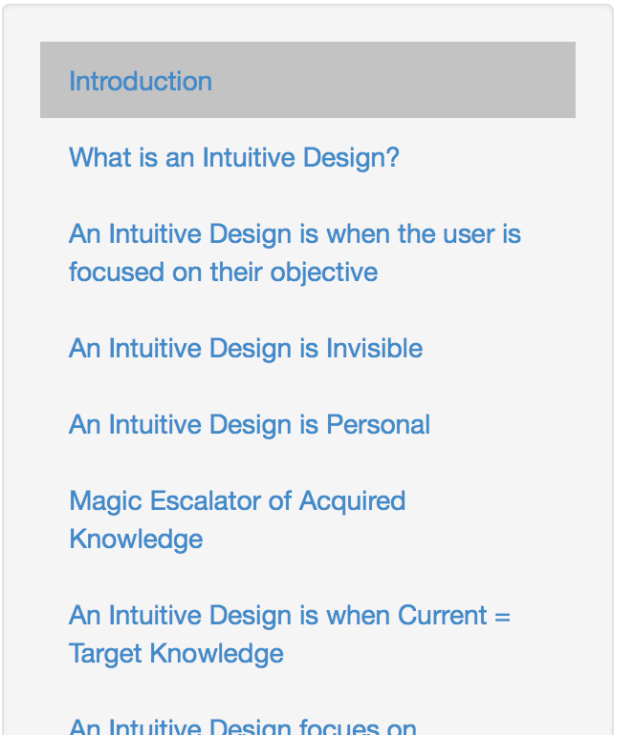
Element	`<combo-player>`
Screenshot	
Description	The ComboPlayer is the core element of an Enhanced Template. It combines both the video stream and slide images in the presentation, and provides controls to allow the viewer to control playback of the media. The position and dimensions of the video and slide elements are determined based on the zoom control, aspect ratios, and layout settings.
Example Code	`<combo-player id="player" playerchrome="combo" allow-edit="playerchrome"></combo-player>`
Generated Code	(Depends on device)
Attributes	**PlayerChrome** ● The PlayerChrome defines the appearance and behavior of the ComboPlayer controls. This setting has no effect when loaded on a device that does not support Flash Player or inline video. ● Attribute name: `playerchrome`

	● Default value: `combo`
	● Possible values:
	  ○ `combo`
	  ■ The player controls appear over the entire ComboPlayer, giving it the appearance of a single, unified media player. While the video is playing, the control bar appears when the user hovers over with the mouse or taps on a touchscreen device. When the user moves the mouse away from the ComboPlayer, or waits several seconds on a touch device, the control bar disappears again.
	  ○ `fixed`
	  ■ The player controls appear over the entire ComboPlayer, giving it the appearance of a single, unified media player. The control bar remains visible at all times.
	  ○ `video`
	  ■ The player controls appear over the video component only. This setting is not generally recommended because the zoom control can cause the video component to be hidden entirely, making it impossible for the user to control the playback.
	  ○ `none`
	  ■ No controls appear. Unless the presentation is configured to start playing automatically, this setting may prevent the user from watching the video.

Notes	● The ComboPlayer may only appear once in a template, so you should not use Responsive Design techniques to hide and show different versions of the ComboPlayer based on screen size.

ChapterNavigator

Element	`<chapter-navigator>`

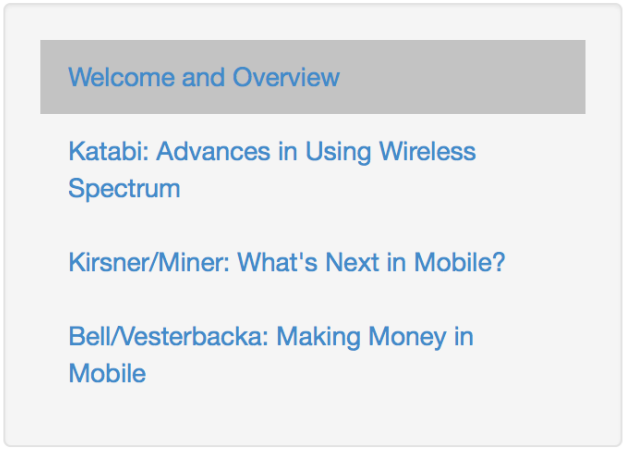
Screenshot	
Description	The ChapterNavigator displays an interactive list of chapter titles. The viewer can click on a chapter in the list to navigate to the start of that chapter in the video timeline. As video continues to play, the current chapter title is highlighted to give the viewer an idea of where they are in the presentation.
Example Code	`<chapter-navigator></chapter-navigator>`

Generated Code	`<ul class="chapter-navigator">`
	`  <li class="chapter-navigator-item-selected">`
	`  <a href="#">{Chapter 1}</a>`
	`  </li>`
	`  <li class="chapter-navigator-item">`
	`  <a href="#">{Chapter 2}</a>`
	`  </li>`
	`  <li class="chapter-navigator-item">`
	`  <a href="#">{Chapter 3}</a>`
	`  </li>`
	`  <!-- ... -->`
	`  <li class="chapter-navigator-item">`
	`  <a href="#">{Chapter N}</a>`
	`  </li>`
	`</ul>`

Attributes	(None)

Notes	

PlaylistNavigator

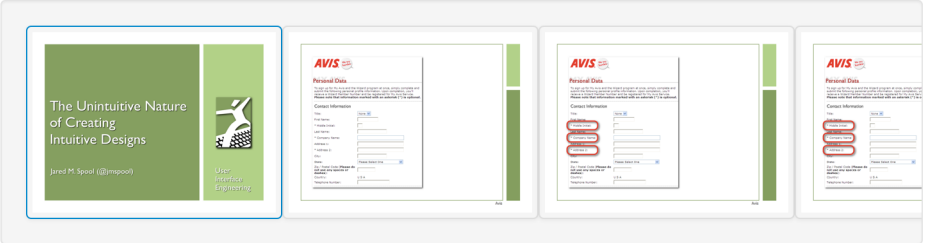
Element	`<playlist-navigator>`
Screenshot	
Description	The PlaylistNavigator displays an interactive list of video titles. The viewer can click on a video title in the list to navigate to the start of that video in the playlist. As presentation continues to play, the current video title is highlighted to give the viewer an idea of where they are in the presentation.
Example Code	`<playlist-navigator></playlist-navigator>`

Generated Code	`<ul class="playlist-navigator">`
	`  <li class="playlist-navigator-item`
	`  playlist-navigator-item-selected">`
	`  <a href="#">{Video 1}</a>`
	`  </li>`
	`  <li class="playlist-navigator-item">`
	`  <a href="#">{Video 2}</a>`
	`  </li>`
	`  <li class="playlist-navigator-item">`
	`  <a href="#">{Video 3}</a>`
	`  </li>`
	`  <!-- ... -->`
	`  <li class="playlist-navigator-item">`
	`  <a href="#">{Video N}</a>`
	`  </li>`
	`</ul>`

Attributes	(None)

Notes	

ThumbnailNavigator

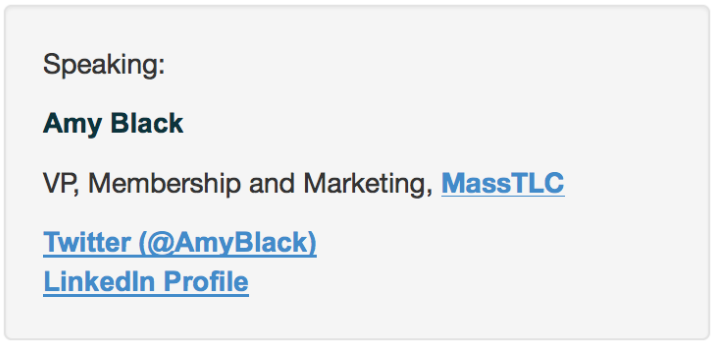
Element	`<thumbnail-navigator>`
Screenshot	
Description	The ThumbnailNavigator displays an interactive list of slide thumbnails. The viewer can click on a slide thumbnail in the list to navigate to the start of that slide in the video. As video continues to play, the current slide is highlighted to give the viewer an idea of where they are in the presentation.
Example Code	`<thumbnail-navigator></thumbnail-navigator>`

Generated Code	`<div class="thumbnail-navigator">`
	`  <img class="thumbnail-navigator-item selected"`
	`  id="0" src="{thumbnail-image-url-1}">`
	`  <img class="thumbnail-navigator-item" id="2"`
	`  src="{thumbnail-image-url-2}">`
	`  <img class="thumbnail-navigator-item" id="3"`
	`  src="{thumbnail-image-url-3}">`
	`  <!-- ... -->`
	`  <img class="thumbnail-navigator-item" id="N"`
	`  src="{thumbnail-image-url-N}">`
	`</div>`

Attributes	(None)
Notes	

Footnotes

Element	`<footnotes>`

Screenshot	
Description	The Footnotes component displays the footnote text associated with the current slide.
Example Code	`<footnotes></footnotes>`
Generated Code	`<div class="footnotes">{Footnote Text}</div>`
Attributes	(None)
Notes	• Footnotes are often formatted with rich text markup, using basic HTML tags. Therefore, it may be important to provide CSS styles to handle elements such as `<a>`, `<b>`, `<i>`, and `<li>`.

Attachments

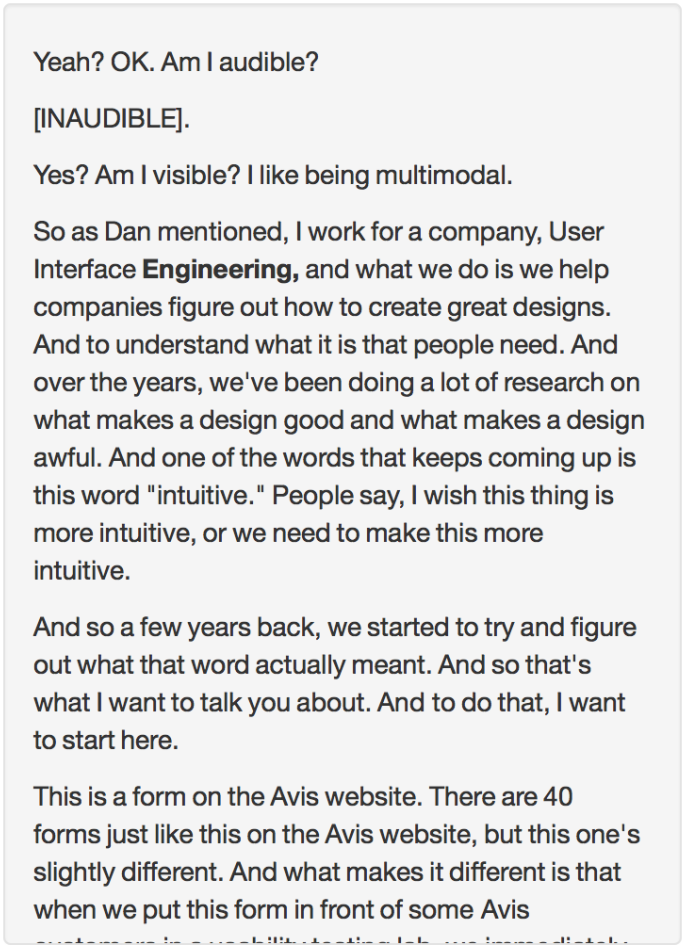
Element	`<attachments>`
Screenshot	
Description	The Attachments component displays the list of attached files or links associated with the presentation. A viewer can click on an item in the lis to download the file or open the link in a new browser tab.
Example Code	`<attachments></attachments>`

Generated Code	`<ul class="attachments">`
	`  <li class="attachments-item">`
	`  <a href="#">{Attachment 1}</a>`

	<pre> <li class="attachments-item"> {Attachment 2} <li class="attachments-item"> {Attachment 3} <!-- ... --> <li class="attachments-item"> {Attachment N} </pre>
Attributes	(None)
Notes	● Unlike the ChapterNavigator and PlaylistNavigator components, there is no selected item in the Attachments list.

Transcript

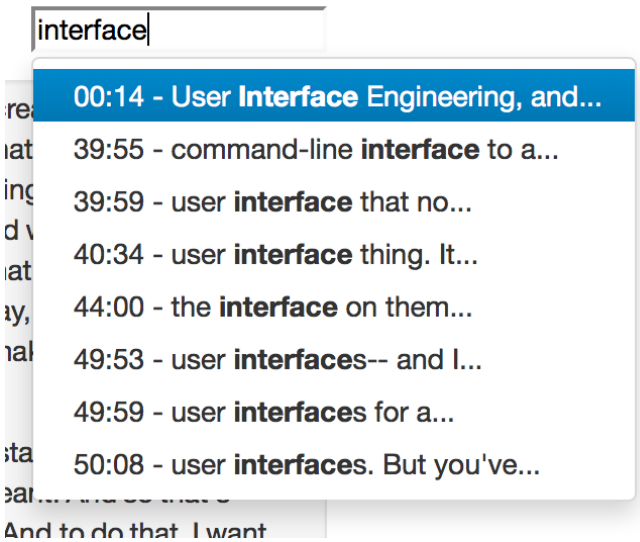
Element	`<transcript>`

Screenshot	
Description	The Transcript component displays interactive, synchronized transcript text of the spoken narration from the playing video. The viewer can click on a word in the transcript text to navigate to the point in the video when that particular word is spoken. As the video continues to play, the word that is being spoken is highlighted in the transcript text.
Example Code	`<transcript></transcript>`

Generated Code	`<div class="transcript">`
	`  <p></p><span data-time="1350"`
	`  class="transcript_1350 selected">{Word`
	`  1}</span><span data-time="2700"`
	`  class="transcript_2700">{Word`
	`  2}</span><p></p><span data-time="3990"`
	`  class="transcript_3990">{Word 3}</span><!--`
	`  ... --><span data-time="4140"`
	`  class="transcript_4140">{Word N}</span>`
	`</div>`

Attributes	(None)
Notes	• The Transcript component can consume a significant amount of browser memory, depending on the length of the video. We recommend only including one Transcript component per template, and hiding it on mobile devices.

TranscriptSearch

Element	`<transcript-search>`
Screenshot	
Description	The TranscriptSearch component is an interactive text input field that allows the viewer to search for a particular word within the transcript text. As the user types her query, the top matching results appear below. A viewer can then click on a result item to navigate to that particular point in the presentation.
Example Code	`<transcript-search></transcript-search>`

Generated Code	`<div class="transcript-search">`
	`<input type="text" placeholder="Search transcript...">`
	`<ul class="typeahead dropdown-menu" style="display: block; top: 43px; left: 162px;">`
	`<li class="active">`
	`<a>00:14 - User <strong>Interface</strong> Engineering, and...</a>`

	<pre>
```
            </li>
            <li>
                  <a>39:55 - command-line
                  <strong>interface</strong> to
                  a...</a>
            </li>
            <li>
                  <a>39:59 - user
                  <strong>interface</strong> that
                  no...</a>
            </li>
            <!-- ... -->
            <li>
                  <a>40:34 - user
                  <strong>interface</strong> thing.
                  It...</a>
            </li>
      </ul>
</div>
```
</pre> |
| **Attributes** | (None) |
| **Notes** | • Although they frequently appear together, the TranscriptSearch component may be used independently of the Transcript component. |

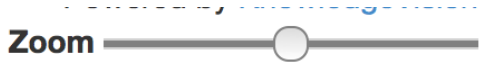
PresentationTitle

Element	`<presentation-title>`
Screenshot	**Jared Spool: The Unintuitive Nature of Creating Intuitive Designs**
Description	The PresentationTitle component displays the title of the presentation.
Example Code	`<presentation-title></presentation-title>`
Generated Code	`<div class="presentation-title">{Presentation Title}</div>`
Attributes	(None)
Notes	

PresentationDescription

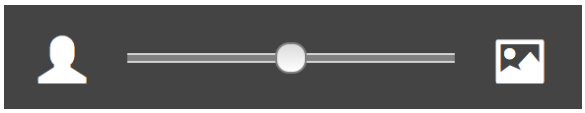
Element	`<presentation-description>`
Screenshot	Recently we hosted a technical conference to give customers a behind-the-scenes look at Alpha Anywhere, our new HTML5 mobile app development and deployment platform. One of the most informative presentations was delivered by Jared Spool, founder of User Interface Engineering, who talked about the importance of an intuitive user experience in business application and mobile website development.
Description	The PresentationDescription component displays the short description of the presentation.
Example Code	`<presentation-description></presentation-description>`
Generated Code	`<p class="presentation-description">{Presentation description text}</p>`
Attributes	(None)
Notes	

ZoomControl

Element	`<zoom-control>`
Screenshot	
Description	The ZoomControl component allows the viewer to control the zoom leve of the video and slide elements within the ComboPlayer. Dragging the slider to the left enlarges the left element, while dragging it to the right enlarges the right element. If the presentation has programmed zooms, the slider will automatically adjust its position to reflect the current zoom level. The ZoomControl is disabled automatically if the presentation playing uses either the “Audio and slides” or “Video only” layout settings
Example Code	`<zoom-control></zoom-control>`
Generated Code	`<input type="range" class="zoom-control">`
Attributes	(None)

Notes	• The ZoomControl uses the HTML5 range input type, which is no fully supported on some older browsers. In this situation, it may appear as a text input instead of a slider control. • Most templates include a ZoomControl, but you may choose to hide it or remove it to prevent the viewer from having this control

ZoomLevel (Zoom Button)

Attribute	`zoom-level`
Screenshot	
Description	Unlike other components, the ZoomLevel attribute does not generate a new element on the page, but instead turns whatever element it is applied to into a Zoom Button component. The Zoom Button component allows the viewer to set a specific zoom level for the video and slide elements within the ComboPlayer. The ZoomLevel attribute can be applied to any HTML element that listens for click events, such as the `<a>`, `<div>`, `<img>`, or `<span>` element. The value of the ZoomLevel attribute determines the zoom level that will be applied to the ComboPlayer. Attribute values may rang from `-1` to show only the video element to `1` to show only the slide element. A value of `0` will show both video and slide elements in equal proportions. Zoom Button components are automatically disabled if the presentation playing uses either the “Audio and slides” or “Video only” layout settings

Example Code	`<!-- Example #1: Set the zoom level to -1 -->`
	`<a href="#" zoom-level="-1">Show Video</a>`
	
	`<!-- Example #2: Set the zoom level to 0 -->`
	`<img src="img/reset.png" zoom-level="0">`
	
	`<!-- Example #3: Set the zoom level to 0.75 -->`
	`<div zoom-level="0.75"><h2>Enlarge Slides</h2></div>`
	
	`<!-- Example #4: Set the zoom level to -0.25 -->`
	`<span zoom-level="-0.25">Enlarge Video</span>`

	`<!-- Example #5: Set the zoom level to 1 -->` `<span zoom-level="1">Hide Video</span>`
Generated Code	`N/A (The HTML code is not changed)`
Notes	• Zoom Buttons may be used in conjunction with a Zoom Control to provide easier click targets, or they may be used independently without a Zoom Control

Fullscreen (Fullscreen Button)

Attribute	`fullscreen`
Screenshot	
Description	Similar to the ZoomLevel attribute, the Fullscreen attribute does not generate a new element on the page, but instead turns whatever element it is applied to into a Fullscreen Button component. The Fullscreen Button component allows the viewer to launch a specified region of the template into fullscreen mode. The Fullscreen attribute can be applied to any HTML element that listen for click events, such as the `<a>`, `<div>`, `<img>`, or `<span>` element. The value of the Fullscreen attribute identifies the `id` of the HTML element that should be set as the container for the fullscreen view. If the Fullscreen attribute is not assigned any value, the default behavior is to open the ComboPlayer element fullscreen. Fullscreen Button components are automatically hidden if the browser does not support the HTML5 Full Screen API or the presentation is embedded inside of an `IFRAME` element that does not have the `AllowFullScreen` attribute enabled.

Example Code	`<!-- Example #1: Open an element with id "mainContainer" in fullscreen mode -->`
	`<a href="#" fullscreen="mainContainer">Fullscreen</a>`
	
	`<!-- Example #2: Open the ComboPlayer in fullscreen mode -->`
	`<input type="button" value="Fullscreen" fullscreen>`

Generated Code	`N/A (The HTML code is not changed)`

Notes	• For full browser support details, please visit http://caniuse.com/#feat=fullscreen

Getting Started Tutorial

To help you get started building your first Enhanced Template, we've put together a tutorial that steps you through the process. The tutorial is broken into a series of examples that progressively build on one another until you have created a fully functional template. The tutorial assumes you already have experience with HTML and CSS.

You can find the full code examples and other resources referenced in this tutorial on Github: https://github.com/Knowledgevision/EnhancedTemplatesGuide

Example 01 - "Hello World!"

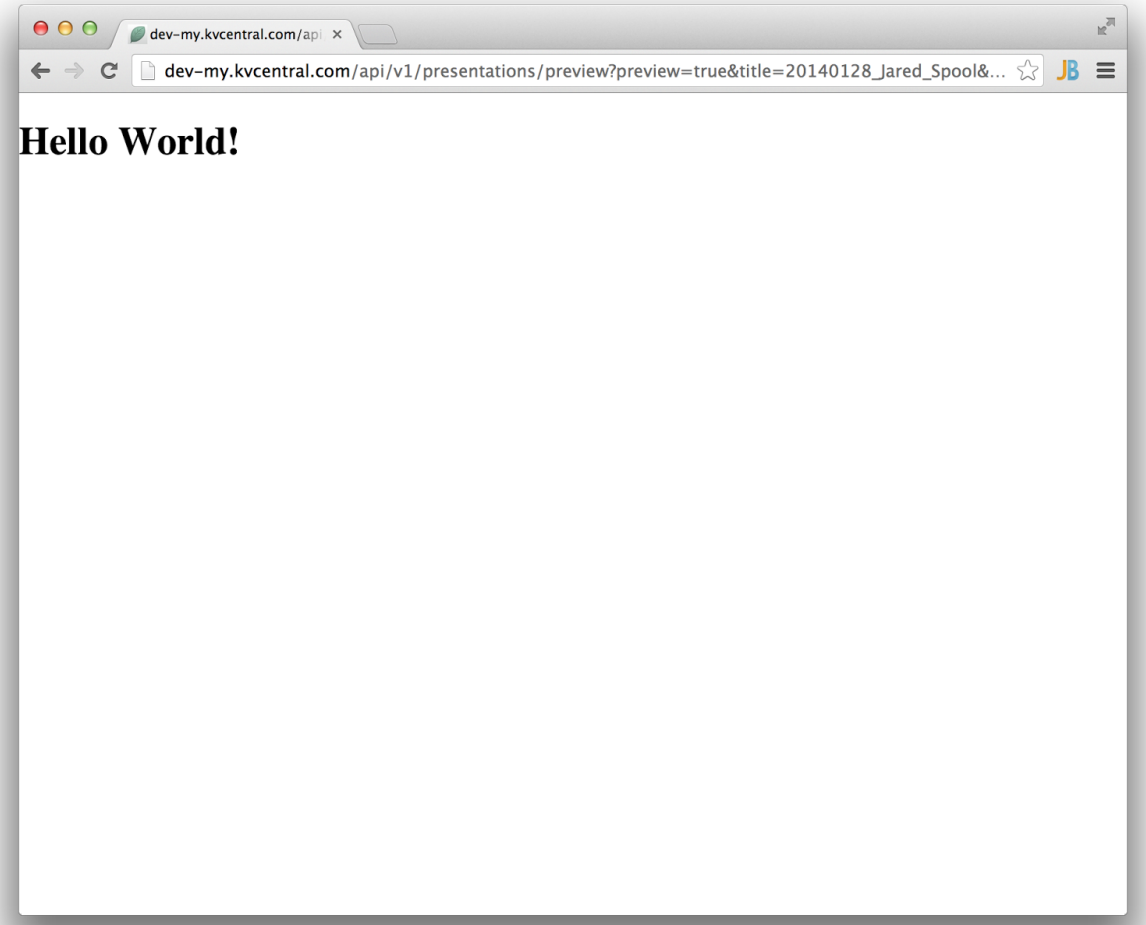
All good programming tutorials begin with the same basic example: demonstrating how to print the words "Hello World!" to the screen. In this case, the goal is simply to show that an Enhanced Template can be nothing more than some basic HTML code.

Here is the code for the Example 01:

example-01.html

```
<!DOCTYPE html>
<html>
<head>
      <title></title>
</head>
<body>

<h1>Hello World!</h1>

</body>
</html>
```

Here is a screenshot of a published presentation using the Example 01 template:



It's not very exciting, and it doesn't even show the video and slides for the presentation! But it demonstrates that there is no minimum requirement for an Enhanced Template. You can start with basic HTML, and then build your way up from there.

Example 02 - The ComboPlayer

The next step is add the core of any online presentation: the video and slides. In Enhanced Templates, both of these elements are combined in a single component called the ComboPlayer. The ComboPlayer serves as the heart of the template, while most other elements are supplementary to it.

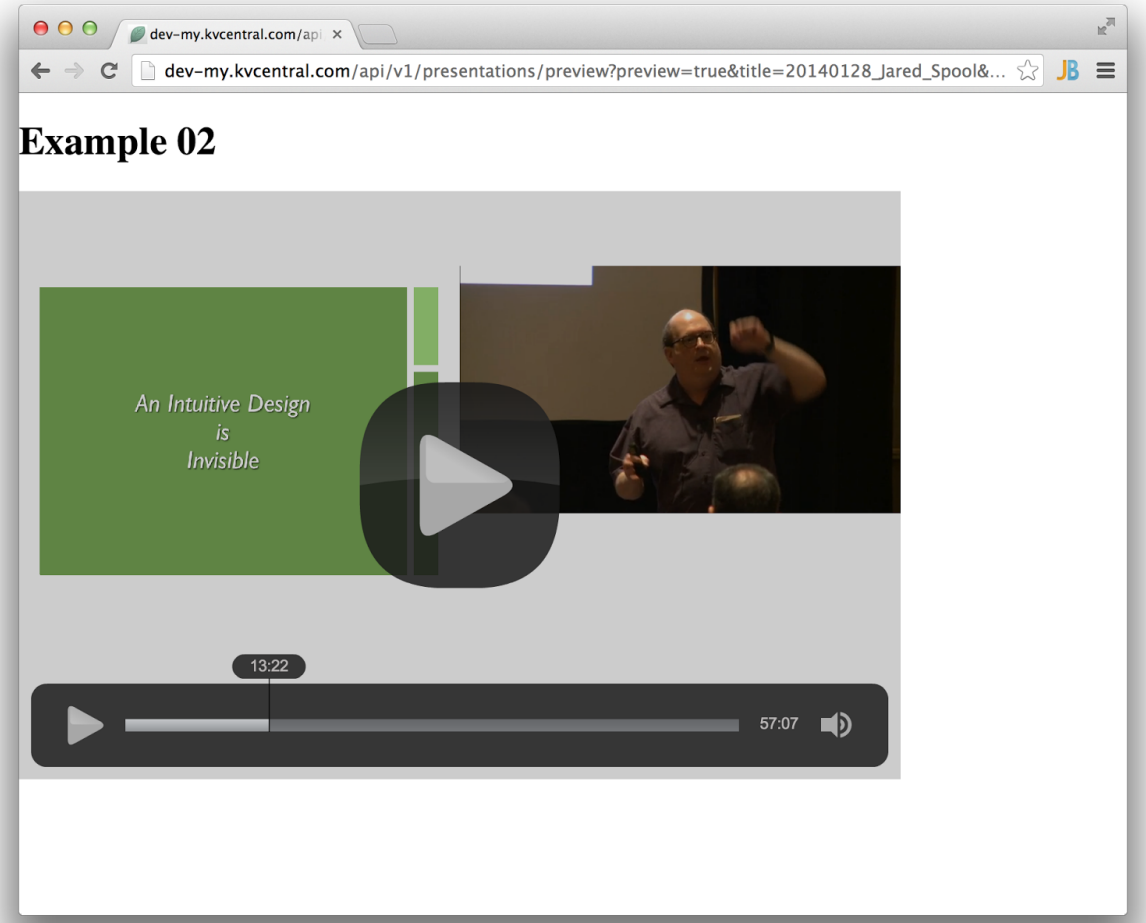
In this example, I want contain the ComboPlayer to fixed size: 720 x 480 pixels. To do this, I wrap the `<combo-player>` element inside of a `<div>` that has fixed dimensions, using inline styles. Notice also that the `position` property is set to `relative` on the container. Elements within the ComboPlayer use absolute positioning, so it is important to set the container to use relative positioning to give it scope.

Here is the code for the Example 02:

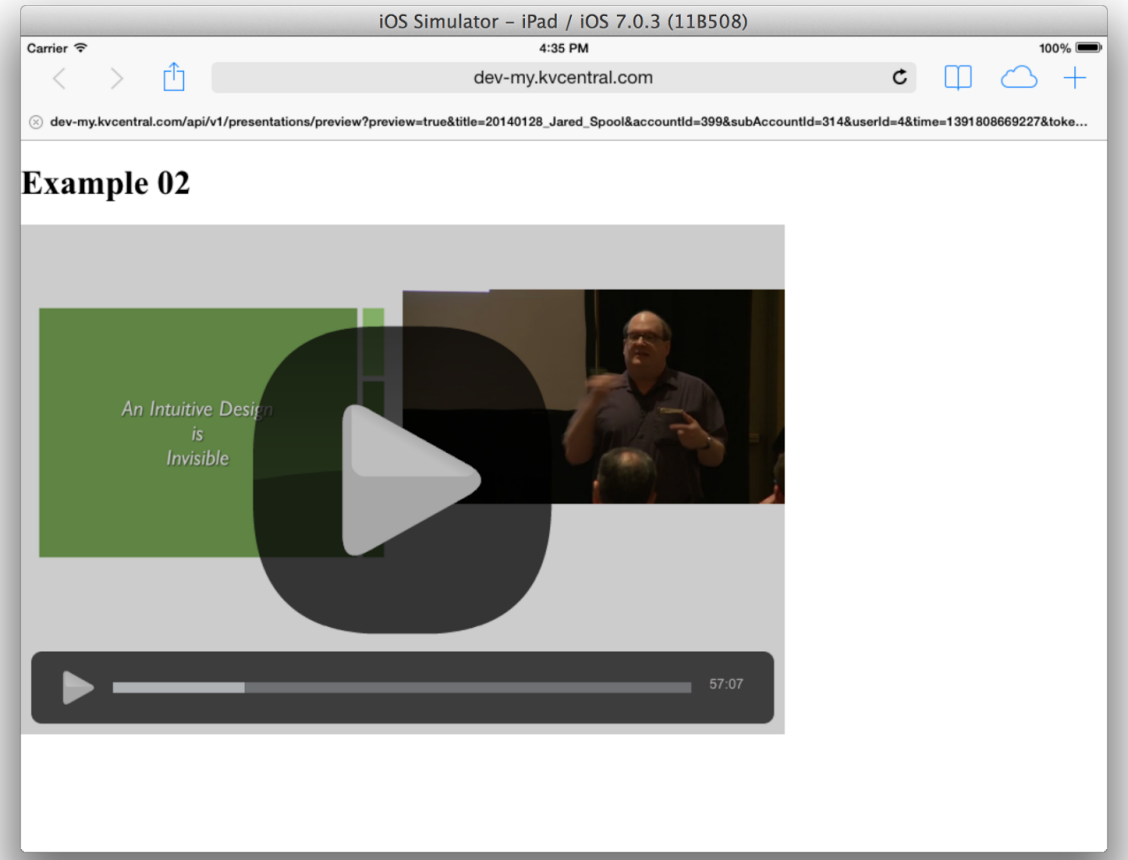
example-02.html

```
<!DOCTYPE html>
<html>
<head>
      <title></title>
</head>
<body>

<h1>Example 02</h1>

<div style="width: 720px; height: 480px; position: relative">

      <combo-player></combo-player>

</div>

</body>
</html>
```

Here is a screenshot of a published presentation using the Example 02 template in a desktop browser:



Here is a screenshot of the same presentation in the iPad Safari browser:



Example 03 - Sizing the ComboPlayer

In most cases, you will want the ComboPlayer to responsively stretch to fit within the width of its containing element, without having to set fixed dimensions. Unlike most HTML elements, except for the `<video>` element, you will typically want to keep the aspect ratio of the ComboPlayer consistent as it stretches to different widths. We recommend a 16:9 aspect ratio, which is standard for widescreen video, but you could optionally use 4:3 or something else.

HTML and CSS don't have a simple built-in mechanism for setting the aspect ratio of an element, so we use a CSS trick that's commonly used for the `<video>` element. We won't get into the specifics, but it essentially requires wrapping the element inside of two containing `<div>` tags. If you're interested, you can read a full explanation here: http://alistapart.com/article/creating-intrinsic-ratios-for-video

We've now added a CSS `<style>` block to the `<head>` of the document to contain the CSS required for the aspect ratios.

To demonstrate how the ComboPlayer resizes dynamically, we've added in container element and set the size to 50% of the browser width.

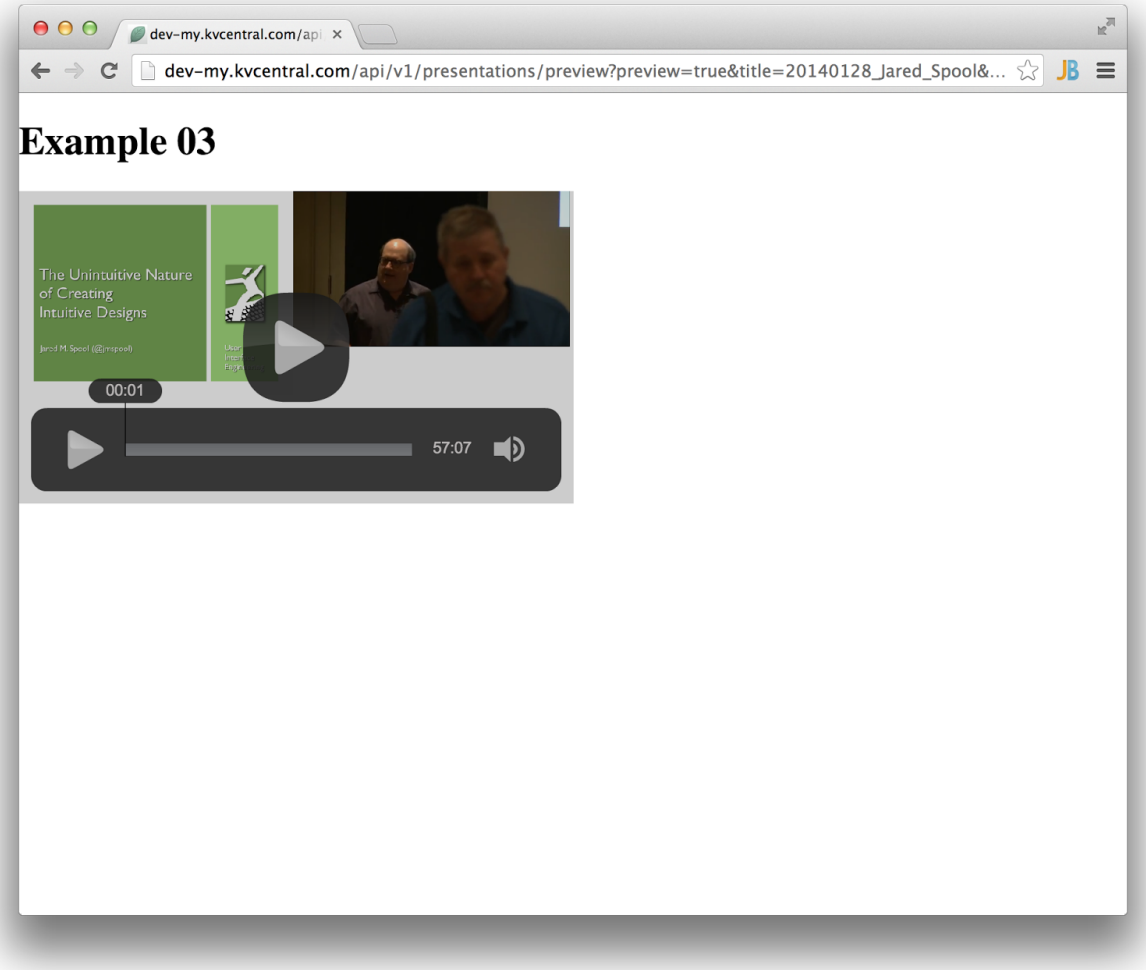
Here is the code for the Example 03:

example-03.html

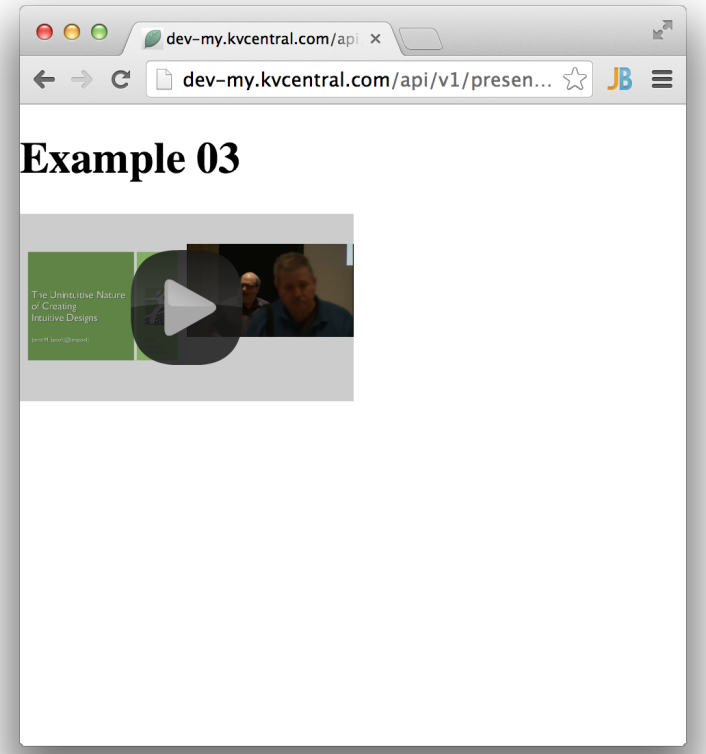
```
<!DOCTYPE html>
<html>
<head>
	<title></title>

	<style>

		.player-wrapper-outer {
			position: relative;
			padding-bottom: 56.25%;
			height: 0;
		}

		.player-wrapper-inner {
			position: absolute;
			top: 0;
			left: 0;
			width: 100%;
			height: 100%;
		}

	</style>

</head>
<body>

<h1>Example 03</h1>

<div style="width: 50%;">

	<div class="player-wrapper-outer">

		<div class="player-wrapper-inner">

			<combo-player></combo-player>

		</div>

	</div>
```

```
</div>

</body>
</html>
```

Here is a screenshot of a published presentation using the Example 03 template in a desktop browser. Notice that it stretches to fill 50% of the width of the browser, but the aspect ratio remains 16:9:



Here is a screenshot of the same presentation with browser narrowed. The ComboPlayer has resized proportionally to remain 50% of the browser window width, but the aspect ratio has remained constant:



Example 04 - Styling the ComboPlayer

The ComboPlayer has a transparent background, allowing you specify the background color, image, or effect you would like to appear behind the video and slides. Since the ComboPlayer is treated as a single unified media player, we recommend choosing a background that gives the impression that the video and slides are not floating in empty space. In our standard templates, we use a gray background with a gradient shadow effect. If you'd prefer to keep things simple, we would recommend a simple black background: `background-color: black;`

Here is the code for the Example 04:

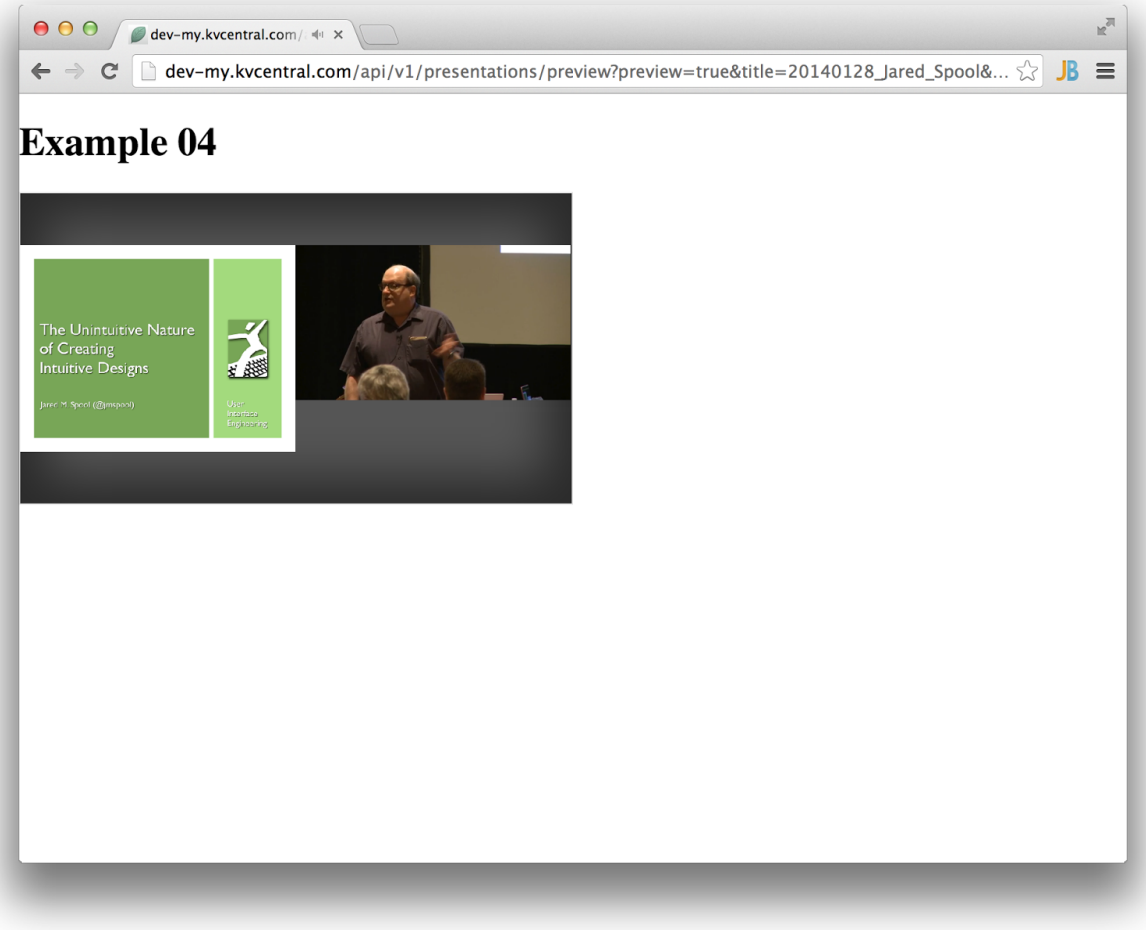
example-04.html

```
<!DOCTYPE html>
<html>
<head>
      <title></title>

      <style>
```

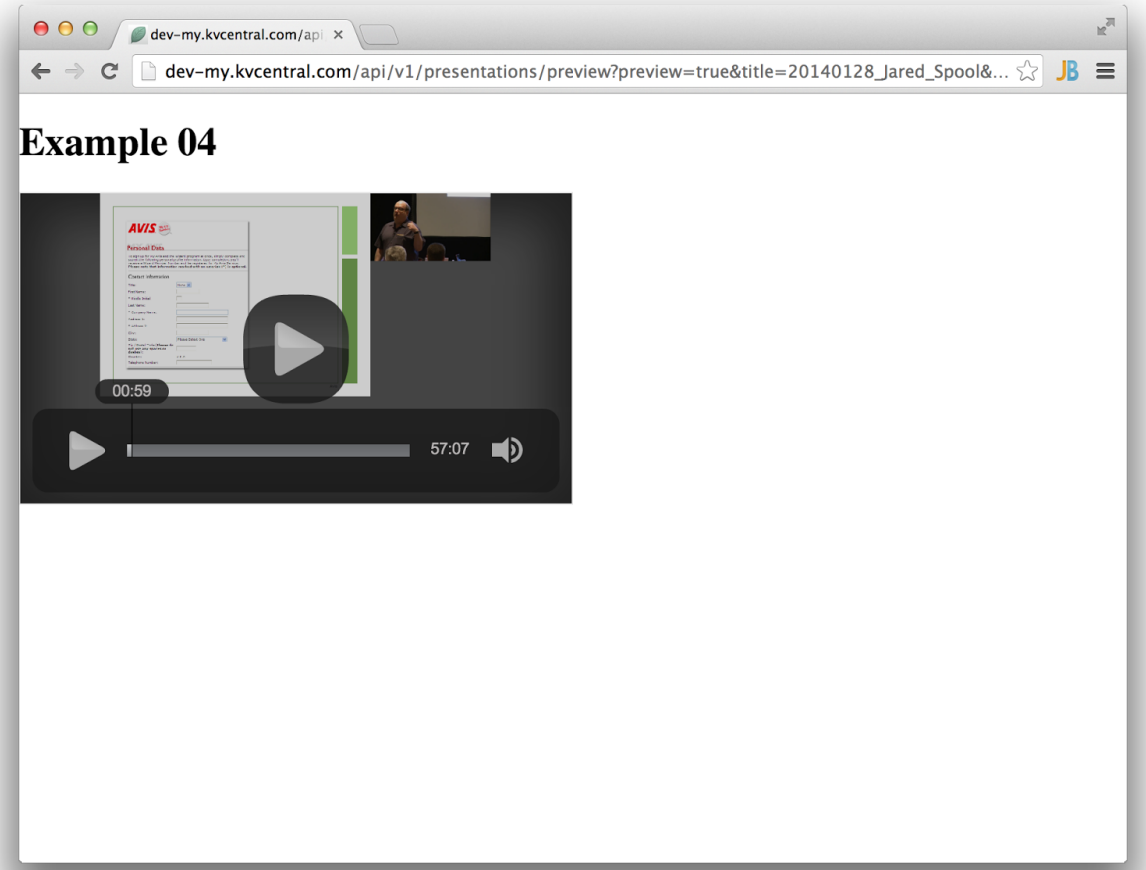
```
            .player-background {
                  border: 1px solid #e3e3e3;
                  background: #555;
                  border-radius: 0;
                  box-shadow: 0 0 60px 20px rgba(0,0,0,0.5) inset;
                  margin: 10px 0;
                  padding: 0;
                  overflow: hidden;
            }

            .player-wrapper-outer {
                  position: relative;
                  padding-bottom: 56.25%;
                  height: 0;
            }

            .player-wrapper-inner {
                  position: absolute;
                  top: 0;
                  left: 0;
                  width: 100%;
                  height: 100%;
            }

      </style>

</head>
<body>

<h1>Example 04</h1>

<div style="width: 50%;">

      <div class="player-background">

            <div class="player-wrapper-outer">

                  <div class="player-wrapper-inner">

                        <combo-player></combo-player>

                  </div>

            </div>

      </div>
```

```
</div>

</body>
</html>
```

Here is a screenshot of a published presentation using the Example 04 template in a desktop browser:



Here the same presentation paused, with a slightly different zoom level:



Example 05 - Supplementary Components

Now that we've got the video and slides showing up in the template with the ComboPlayer, let's try adding another supplementary interactive component. In this example, we'll add basic chapter navigation, using the ChapterNavigator component. The ChapterNavigator is an interactive list of chapter titles. The viewer can click on a chapter in the list to navigate to the start of that chapter in the video timeline. As video continues to play, the current chapter title is typically highlighted to give the viewer an idea of where they are in the presentation.

We've also added some basic CSS to define a fluid two-column layout. The left column, containing the ComboPlayer, fills 75% of browser width, while the right column, containing the ChapterNavigator, fills the remaining 25%.

Here is the code for the Example 05:

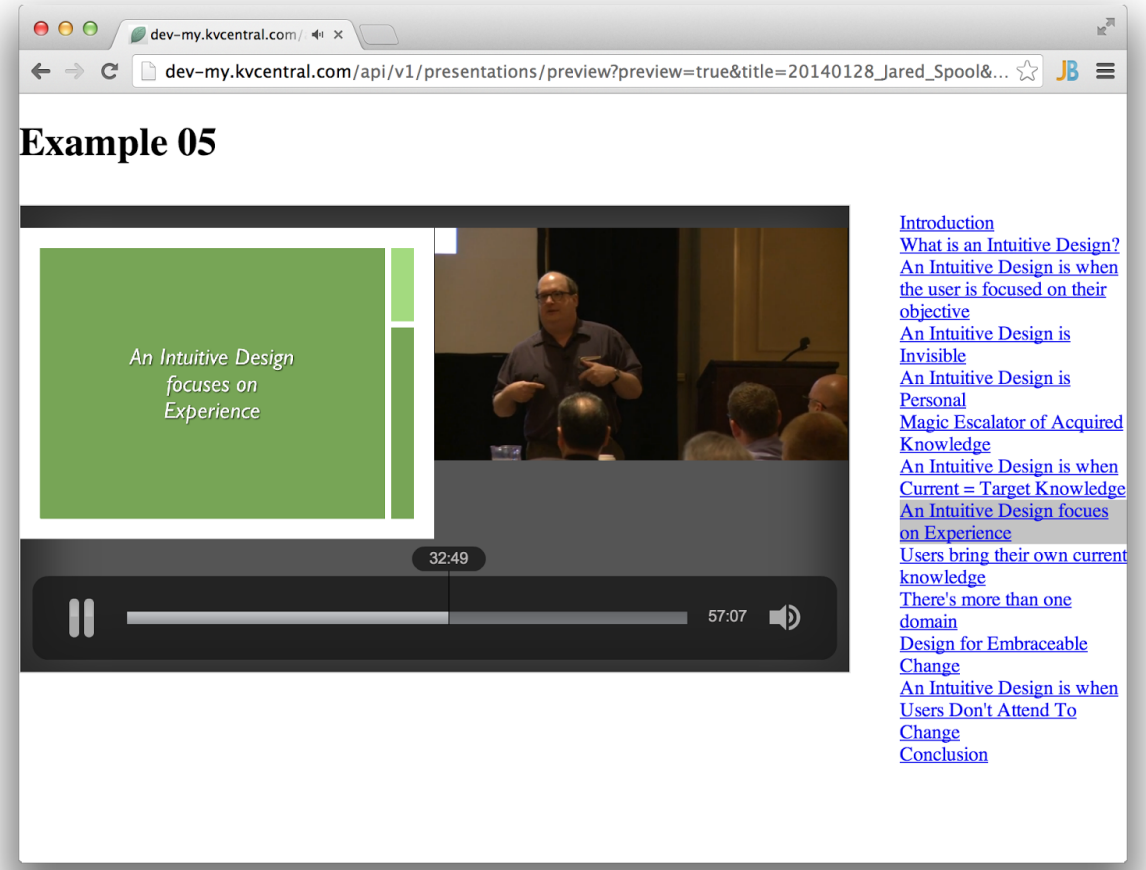
example-05.html

```
<!DOCTYPE html>
```

```
<html>
<head>
	<title></title>

	<style>

		.column {
			display: inline-block;
			float: left;
		}

		.player-background {
			border: 1px solid #e3e3e3;
			background: #555;
			border-radius: 0;
			box-shadow: 0 0 60px 20px rgba(0,0,0,0.5) inset;
			margin: 10px 0;
			padding: 0;
			overflow: hidden;
		}

		.player-wrapper-outer {
			position: relative;
			padding-bottom: 56.25%;
			height: 0;
		}

		.player-wrapper-inner {
			position: absolute;
			top: 0;
			left: 0;
			width: 100%;
			height: 100%;
		}

	</style>

</head>
<body>

<h1>Example 05</h1>

<div class="column" style="width: 75%;">

	<div class="player-background">

		<div class="player-wrapper-outer">
```

```
                <div class="player-wrapper-inner">

                     <combo-player></combo-player>

                </div>

           </div>

     </div>

</div>

<div class="column" style="width: 25%;">
     <chapter-navigator></chapter-navigator>
</div>

</body>
</html>
```

Here is a screenshot of a published presentation using the Example 05 template:



Example 06 - Styling Supplementary Components

The ChapterNavigator component appears as a single element in our HTML, but when the template is rendered, the player generates a list element containing items for each chapter title in the presentation. This generated HTML code looks something like this:

ChapterNavigator Generated Code Sample

```
<ul class="chapter-navigator">
     <li class="chapter-navigator-item-selected">
          <a href="#">{Chapter 1}</a>
     </li>
     <li class="chapter-navigator-item">
          <a href="#">{Chapter 2}</a>
     </li>
     <li class="chapter-navigator-item">
          <a href="#">{Chapter 3}</a>
```

```
      </li>
      ...
      <li class="chapter-navigator-item">
            <a href="#">{Chapter N}</a>
      </li>
</ul>
```

By knowing the structure of the generated HTML code, we can use CSS to style the individual elements. In particular, notice that the chapter item that's currently playing is dynamically tagged with the `chapter-navigator-item-selected` class name, while the others use the `chapter-navigator-item` class. We can use these class names to highlight the current chapter title as the video plays in our CSS.

You can also assign ID and class names to custom components, and these will be applied to the top-level generated element. For example, if we give an ID of `my-chapter-navigator` to our ChapterNavigator component:
`<chapter-navigator id="my-chapter-navigator"></chapter-navigator>`

the generated `<ul>` element will use this ID:
`<ul id="my-chapter-navigator">...</ul>`

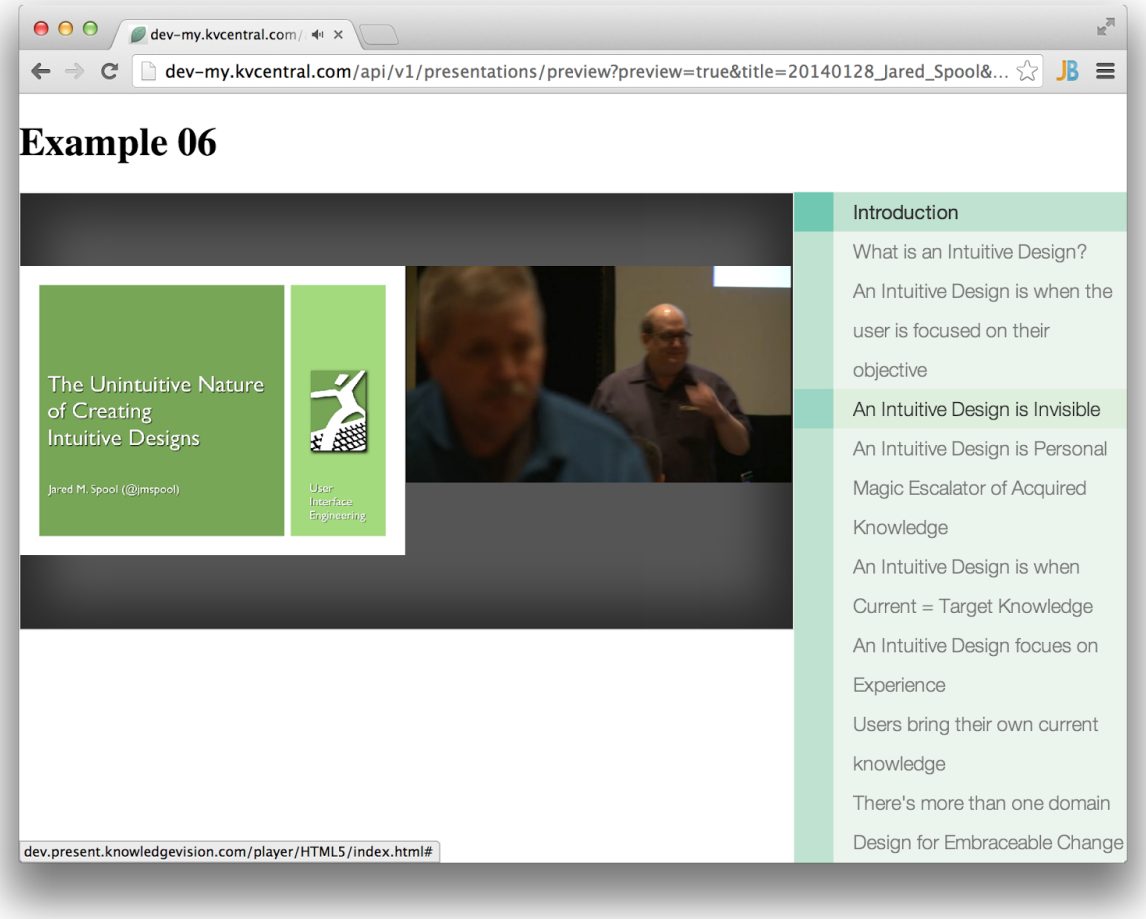
Rather than reproduce the entire document, here is the new CSS code added for the Example 06:

example-06.html

```
#my-chapter-navigator {
      font-family: "Helvetica Neue", Helvetica, sans-serif;
      font-weight: 300;
      line-height: 2em;
      margin: 0;
      padding: 0;
}

#my-chapter-navigator li {
      list-style:none;
      cursor: pointer;
      background-color: #ECF5ED;
      border-color: #C0E2D1;
      border-width: 0 0 0 2em;
      border-style: solid;
      padding-left: 1em;
}
```

```
#my-chapter-navigator a {
     color: #777;
     text-decoration: none;
}

#my-chapter-navigator .chapter-navigator-item:hover {
     background-color:#DFEFDE;
     border-color: #9BD5C3;
}

#my-chapter-navigator .chapter-navigator-item:hover a,
#my-chapter-navigator .chapter-navigator-item-selected a {
     color: #333;
}

#my-chapter-navigator .chapter-navigator-item-selected {
     background-color:#C0E2D1;
     border-color: #70C8B2;
     font-weight: 400;
}

#my-chapter-navigator .chapter-navigator-item-selected:hover {
     background-color:#C0E2D1;
     border-color: #70C8B2;
}
```

Here is a screenshot of a published presentation using the Example 06 template:



Example 07 - Directory Structure

To complete this simple example template, we'll add a linked logo image to the top. There's no special component for this, it's just an image element inside of an anchor:

```
<a href="http://www.knowledgevision.com" target="_blank"><img src="img/KnowledgeVisionLogo.jpg" /></a>
```

However, we will need to store our logo image somewhere. Up until this point, we've been focused on a single HTML file, but a template can be a package of resources that includes HTML, CSS, JavaScript, and image files. So we've created a subdirectory called `img` in our template project folder, and used a relative path reference to the image.

While we're at it, let's also move our styles into its own CSS file. Again, just to keep our template project well organized, we'll store it in another subdirectory called `css`, and then replace it with a `<link>` element in our HTML.

```
<link rel="stylesheet" type="text/css" href="css/example.css">
```

In both cases we've used relative paths to reference the files. However, when the template package is uploaded to our server, our system will actually search for any relative paths in the HTML file and automatically replace them with absolute URLs pointing the uploaded location of files.

Now, our template project file structure looks like this:

```
example-07/
├── css/
│   └── example.css
├── img/
│   └── KnowledgeVisionLogo.jpg
└── index.html
```

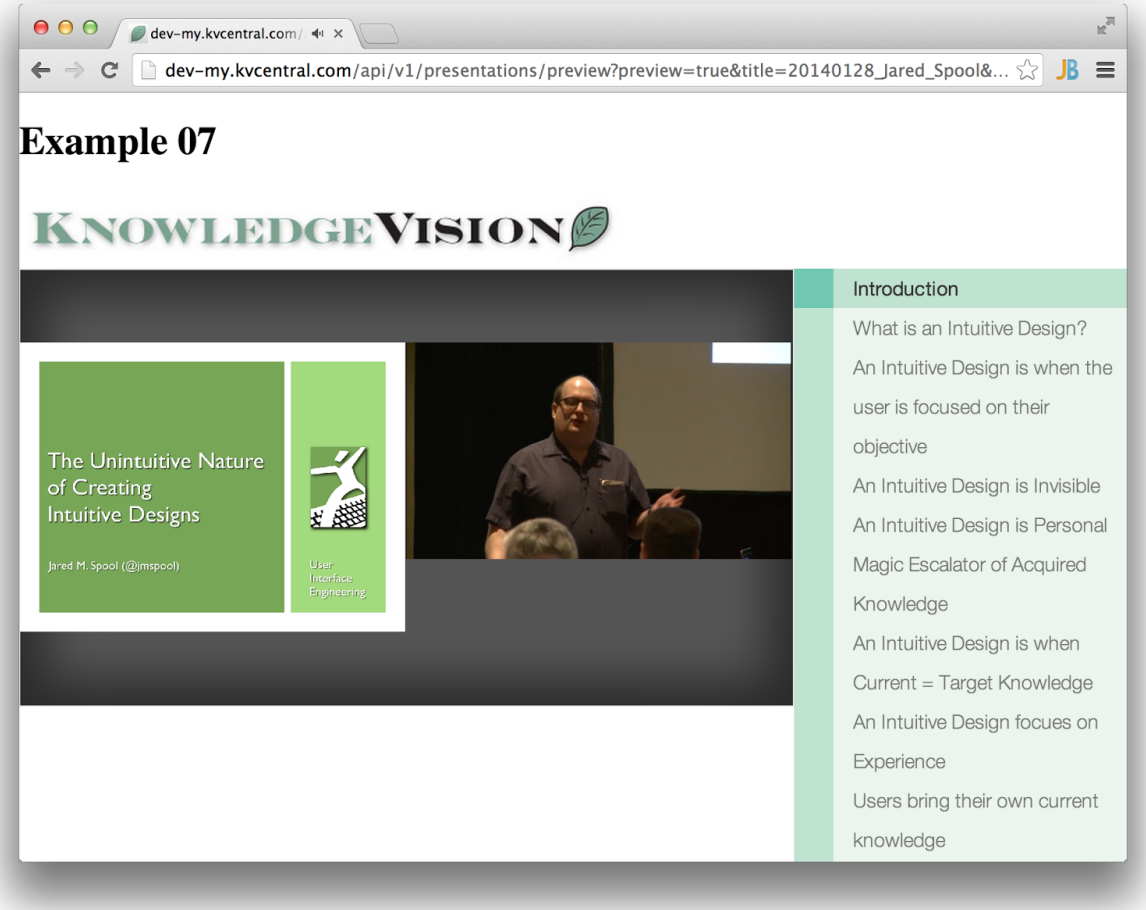
Here is the HTML code for the Example 07:

example-07.html

```
<!DOCTYPE html>
<html>
<head>
     <title></title>

     <link rel="stylesheet" type="text/css" href="css/example.css">

</head>
<body>

<h1>Example 07</h1>

<a href="http://www.knowledgevision.com" target="_blank"><img
src="img/KnowledgeVisionLogo.jpg" /></a>

<div class="column" style="width: 70%;">

     <div class="player-background">

          <div class="player-wrapper-outer">

               <div class="player-wrapper-inner">

                    <combo-player></combo-player>

               </div>
```

```
            </div>

        </div>

</div>

<div class="column" style="width: 30%;">

        <chapter-navigator
id="my-chapter-navigator"></chapter-navigator>

</div>

</body>
</html>
```

Here is a screenshot of a published presentation using the Example 07 template:



Example 08 - Editable Properties

At this point, we've got a decently functional template with a ComboPlayer, ChapterNavigator, and linked logo image. From here, we could continue to add additional components and tweak the appearance to get the look and feel just right. However, one additional consideration we must take is what, if any, control we wish to give to the user who will publish presentations with this template.

With Enhanced Templates, the template designer has complete control over which properties a publisher has permission to customize. You can make almost any HTML attribute or CSS style editable by the publishing user by adding a custom attribute to an element called `allow-edit`. Within the value of this attribute, you list the editable properties that you wish to expose.

The `allow-edit` attribute will accept a list that can include one or more of the following:

- Standard HTML element attributes:
  `<a id="link" href="#" allow-edit="href">...</a>`

- Custom component attributes, such as `playerchrome` on the ComboPlayer:
  `<combo-player id="player" playerchrome="combo" allow-edit="playerchrome"></combo-player>`

- CSS styles, using the format `style.{property}`:
  `<div id="wrapper" allow-edit="style.background">...</div>`

- Text, using the special value `text`:
  `<h3 id="title" allow-edit="text">My Text</h3>`

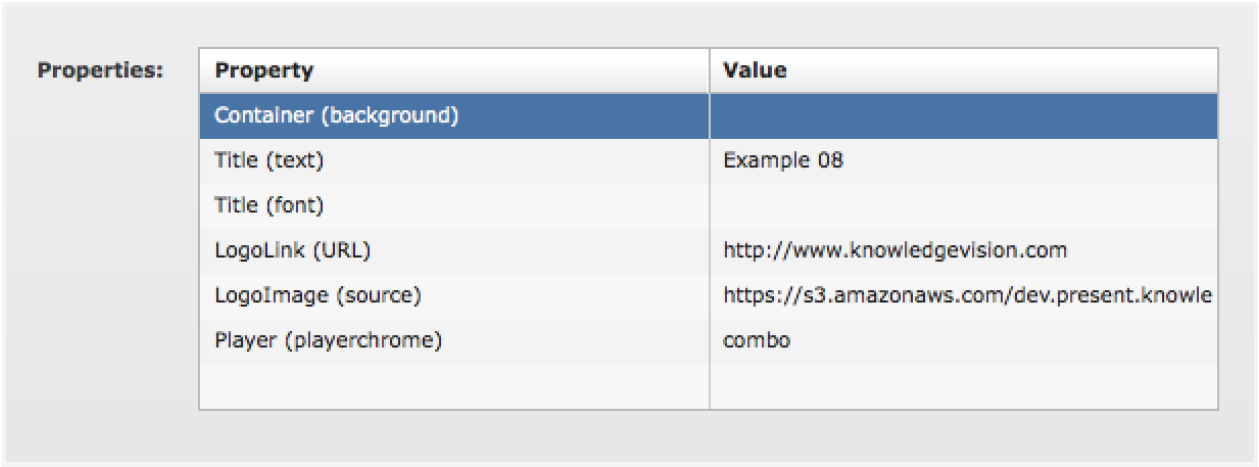
In this example, there are a handful of editable properties we want to expose to the publisher user. In particular, we want to allow the user to swap out the logo image and direct it to a different URL. To accomplish this, we need to give the `<a>` and `<img>` tags IDs to identify them, then add the `allow-edit` attribute to each and list the appropriate editable properties. Here is the resulting code:

```
<a href="http://www.knowledgevision.com" target="_blank"
id="logoLink" allow-edit="href"><img
src="img/KnowledgeVisionLogo.jpg" id="logoImage" allow-edit="src"
/></a>
```

We've gone through and added several other editable properties throughout the HTML structure, including the background style, the title text and font, and the ComboPlayer chrome behavior.

The result is that when the publisher user selects our template in KVStudio, she will see these properties listed and be able to change the default values if she so desires.

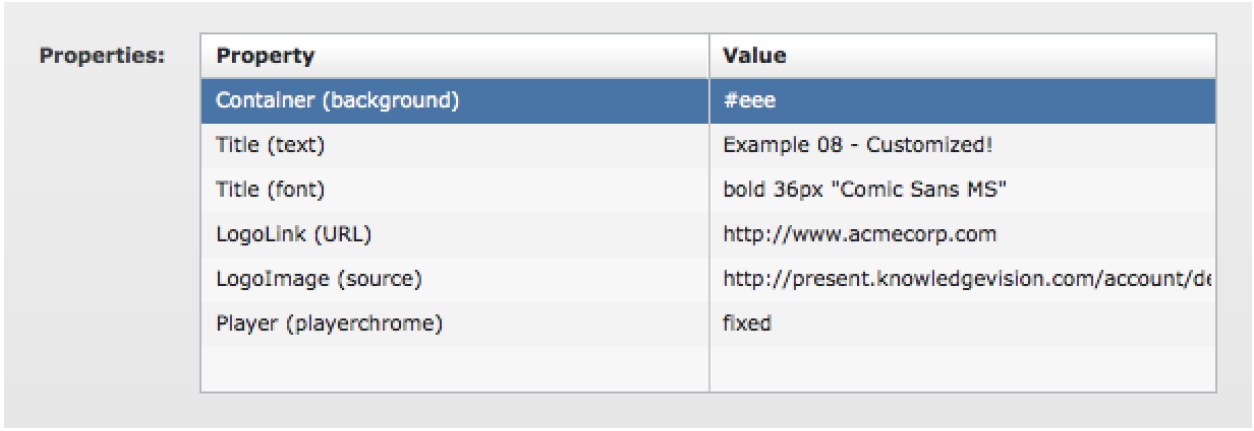
Here is a screenshot of what a KVStudio user would see when selecting this template for a project in Step 4, under the Template Options tab:



Properties:

Property	Value
Container (background)	
Title (text)	Example 08
Title (font)	
LogoLink (URL)	http://www.knowledgevision.com
LogoImage (source)	https://s3.amazonaws.com/dev.present.knowle
Player (playerchrome)	combo

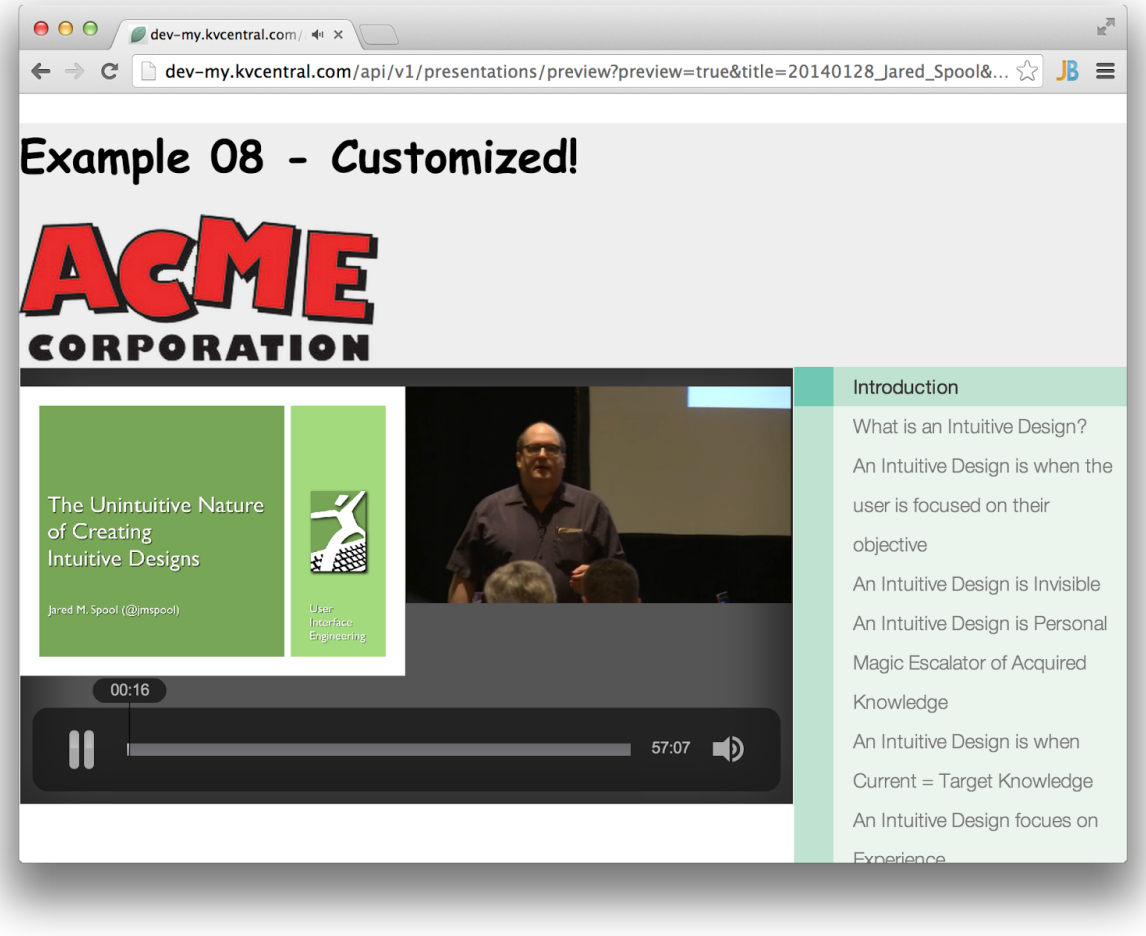
The user could then edit these properties as desired (though maybe not advised):



Properties:

Property	Value
Container (background)	#eee
Title (text)	Example 08 - Customized!
Title (font)	bold 36px "Comic Sans MS"
LogoLink (URL)	http://www.acmecorp.com
LogoImage (source)	http://present.knowledgevision.com/account/de
Player (playerchrome)	fixed

Here is a screenshot of a published presentation using the Example 08 template with the custom properties specified above:



Here is the code for the Example 08:

example-08.html

```
<!DOCTYPE html>
<html>
<head>
      <title></title>

      <link rel="stylesheet" type="text/css" href="css/example.css">

</head>
<body>

<div id="container" allow-edit="style.background">

      <h1 id="title" allow-edit="text,style.font">Example 08</h1>
```

```
     <a href="http://www.knowledgevision.com" target="_blank"
id="logoLink" allow-edit="href"><img
src="img/KnowledgeVisionLogo.jpg" id="logoImage" allow-edit="src"
/></a>

     <div class="column" style="width: 70%;">

     <div class="player-background">

          <div class="player-wrapper-outer">

               <div class="player-wrapper-inner">

                    <combo-player id="player" playerchrome="combo"
allow-edit="playerchrome"></combo-player>

               </div>

          </div>

     </div>

     </div>

     <div class="column" style="width: 30%;">

          <chapter-navigator
id="my-chapter-navigator"></chapter-navigator>

     </div>
</div>

</body>
</html>
```

Example 09 - Design Helper

The KnowledgeVision Player automatically replaces your component tags with HTML elements when it loads your Enhanced Template at runtime. As a result, it can be difficult to test your design without uploading it and applying it to a published presentation.

To make designing Enhanced Templates easier, we've created a simple Design Helper utility script that you can temporarily include in your template during the design process. The script simulates the component replacement process and generates sample HTML elements that match the format used by the player.

The script depends on JQuery, but the player also loads JQuery at runtime, so this too simulates the experience. The script also loads the default CSS file used by the player, which may affect your own stylesheets.

We have hosted a copy of the Design Helper script, so including it in your template HTML is as simple as adding two lines of code to the `<head>` element:

Design Helper Script Sample

```
<!-- Start: Design Helper -->
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script
src="http://dev.knowledgevision.com/scripts/design-helper-v1.js"></
script>
<!-- End: Design Helper -->
```

When you are ready to upload your finished template, simply remove these lines or comment them out:

Design Helper Script Sample - Commented Out

```
<!-- Start: Design Helper -->
<!--<script
src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script
src="http://dev.knowledgevision.com/scripts/design-helper-v1.js"></
script>-->
<!-- End: Design Helper -->
```

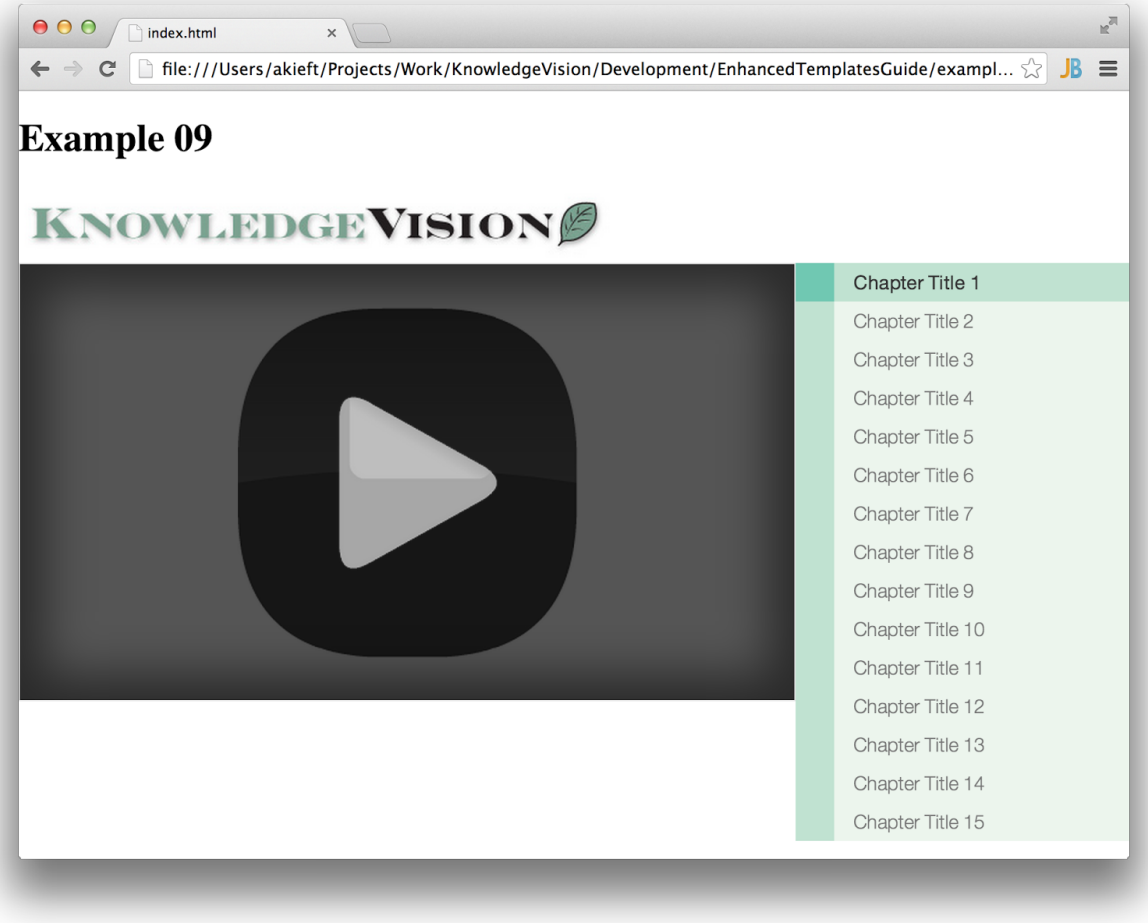
Here is the full HTML code for the Example 09, with the Design Helper script included for local testing:

example-09.html

```
<!DOCTYPE html>
<html>
<head>
     <title></title>

     <!-- Start: Design Helper -->
     <script
     src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
     <script
     src="http://dev.knowledgevision.com/scripts/design-helper-v1.j
     s"></script>
```

```
     <!-- End: Design Helper -->

     <link rel="stylesheet" type="text/css" href="css/example.css">

</head>
<body>

<h1>Example 09</h1>

<a href="http://www.knowledgevision.com" target="_blank"><img
src="img/KnowledgeVisionLogo.jpg" /></a>

<div class="column" style="width: 70%;">

     <div class="player-background">

          <div class="player-wrapper-outer">

               <div class="player-wrapper-inner">

                    <combo-player></combo-player>

               </div>

          </div>

     </div>

</div>

<div class="column" style="width: 30%;">

     <chapter-navigator
id="my-chapter-navigator"></chapter-navigator>

</div>

</body>
</html>
```

Here is a screenshot of the Example 09 template running locally, using the Design Helper script:



Responsive vs. Elastic vs. Fixed Size

As you build an Enhanced Template, you will need to consider the sizing constraints that may be placed on it when a publisher embeds the presentation on his website. In some situations, you may know in advance that all presentations using your template will always be embedded on a page with a rigid layout, in which case you can optimize your design by giving your template fixed dimensions. In other cases, you may not know where the presentations will be embedded, so you will need to design your template to stretch elastically to fit the available space. A final possibility is that your template will be used in presentations that fill the entire page, in which case it will need to dynamically respond to the dimensions of the browser window or device screen.

Bootstrap

Most of KnowledgeVision's standard Enhanced Templates use Twitter's Bootstrap framework to provide a consistent look and feel. Bootstrap provides a simple, responsive grid layout system, as well as a clean, modern stylesheet for your standard HTML elements. It also includes an extensive set of user interface components like tabs, dropdowns, and navbars that can make your template feel very dynamic. For more information about Twitter Bootstrap, visit http://getbootstrap.com

JQuery

If you interested in building Enhanced Templates that offer even more interactive capabilities than are provided by standard components alone, you will probably need to use JavaScript programming. Because the KVPlayer already uses JavaScript to enable the functionality of the template, you will need to be careful to scope your code so as not to interfere with the core behavior. However, you can also benefit from the fact that the KVPlayer already uses the JQuery library, which makes DOM manipulation significantly easier. As a result, you do not need to add JQuery directly to your template HTML, except possibly while you are testing your code locally. For more information about JQuery, visit http://jquery.com/

Deploying Enhanced Templates

Once your template is built, the next step is deploying it to your KnowledgeVision account. Enhanced Templates are assets that are available to all users of your account to apply to their presentations in KVStudio.

Preview Image

When you upload an Enhanced Template to your account, our system automatically generates a preview image of the template to display in the Enhanced Template Management Module and in KVStudio. This preview image allows users to see what the template is going to look like before they publish a presentation with it.

If you would prefer to supply your own preview image rather than allow our system to generate one, you can can include it in your template package. We recommend taking a screenshot of your template while using the Design Helper script so that users can see what it will look like with sample content. We also recommend that you size the image to be 720 pixels wide by 540 pixels tall to prevent it from appearing squished or squeezed in KVStudio.

In order for our system to recognize your preview image file, it must be:

- Saved in PNG format
- Named `preview.png`

- Included in the root directory of your package.

Package Structure

Each Enhanced Template is initially loaded from an HTML file, but this file may reference other stylesheets, images, and scripts. You must package all of these files into a single ZIP file that you upload to your account. Our system then unpackages the ZIP file and moves the individual files to our web server for hosting.

During this process, our system will also search for any relative file paths in the HTML file and automatically replace them with absolute URLs pointing the locations of the uploaded files. This is necessary because the base URL of the player does not point to the HTML template file, and therefore relative paths in the HTML will break. However, relative paths used in external CSS files should continue to work as expected.

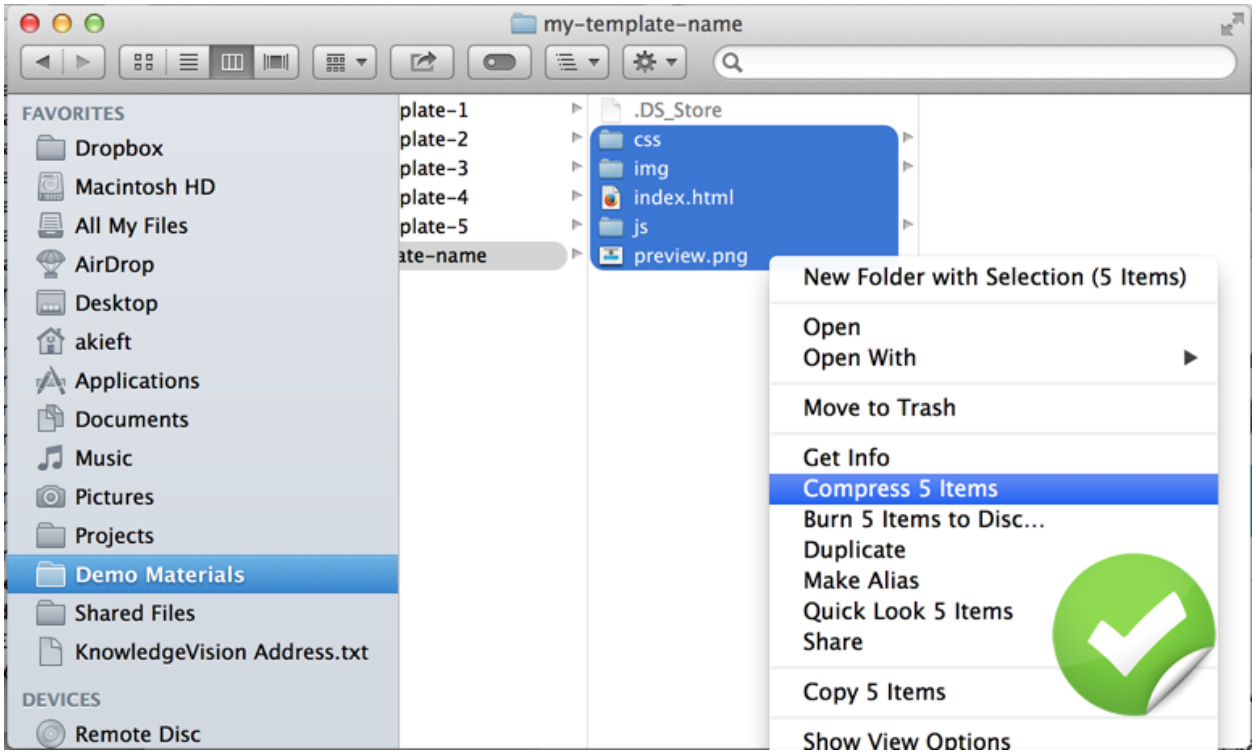
Any files that you choose to include in your template should be organized into three subdirectories: `css`, `img`, and `js`. Other subdirectories may be ignored by our system. Although it is not recommended, files other than the HTML file and preview image may also be included in the root directory of the package.

An example template project file structure looks like this:

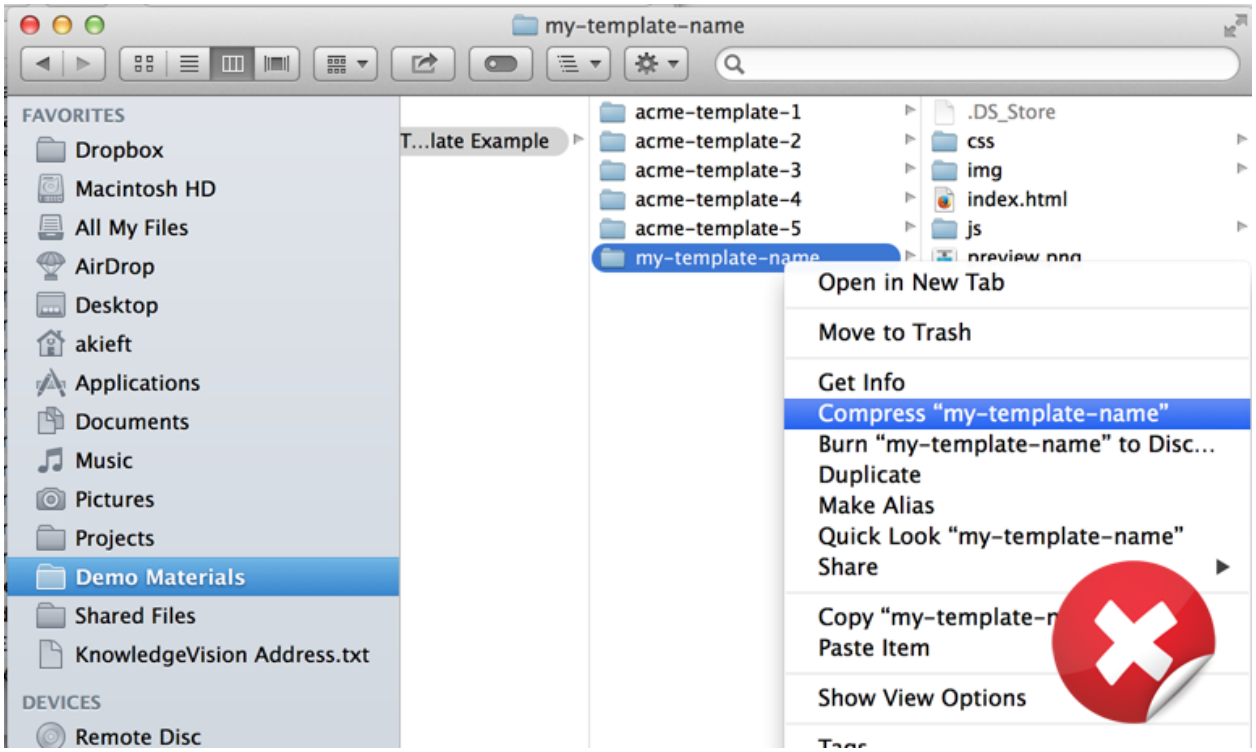
```
my-template-name/
├── css/
│   ├── style-1.css
│   ├── style-2.css
│   └── style-3.css
├── img/
│   ├── image-1.png
│   ├── image-2.png
│   └── image-3.png
├── js/
│   ├── script-1.js
│   ├── script-2.js
│   └── script-3.js
├── index.html
└── preview.png
```

The contents of the template project folder should be compressed into a single ZIP file. On Mac OS X, you must select the group of files within the folder, rather than the folder itself, when you generate the ZIP file.

Correct:


my-template-name
FAVORITES
Dropbox
Macintosh HD
All My Files
AirDrop
Desktop
akieft
Applications
Documents
Music
Pictures
Projects
Demo Materials
Shared Files
KnowledgeVision Address.txt
DEVICES
Remote Disc
plate-1
plate-2
plate-3
plate-4
plate-5
ate-name
.DS_Store
css
img
index.html
js
preview.png
New Folder with Selection (5 Items)
Open
Open With
Move to Trash
Get Info
Compress 5 Items
Burn 5 Items to Disc...
Duplicate
Make Alias
Quick Look 5 Items
Share
Copy 5 Items
Show View Options

Incorrect:

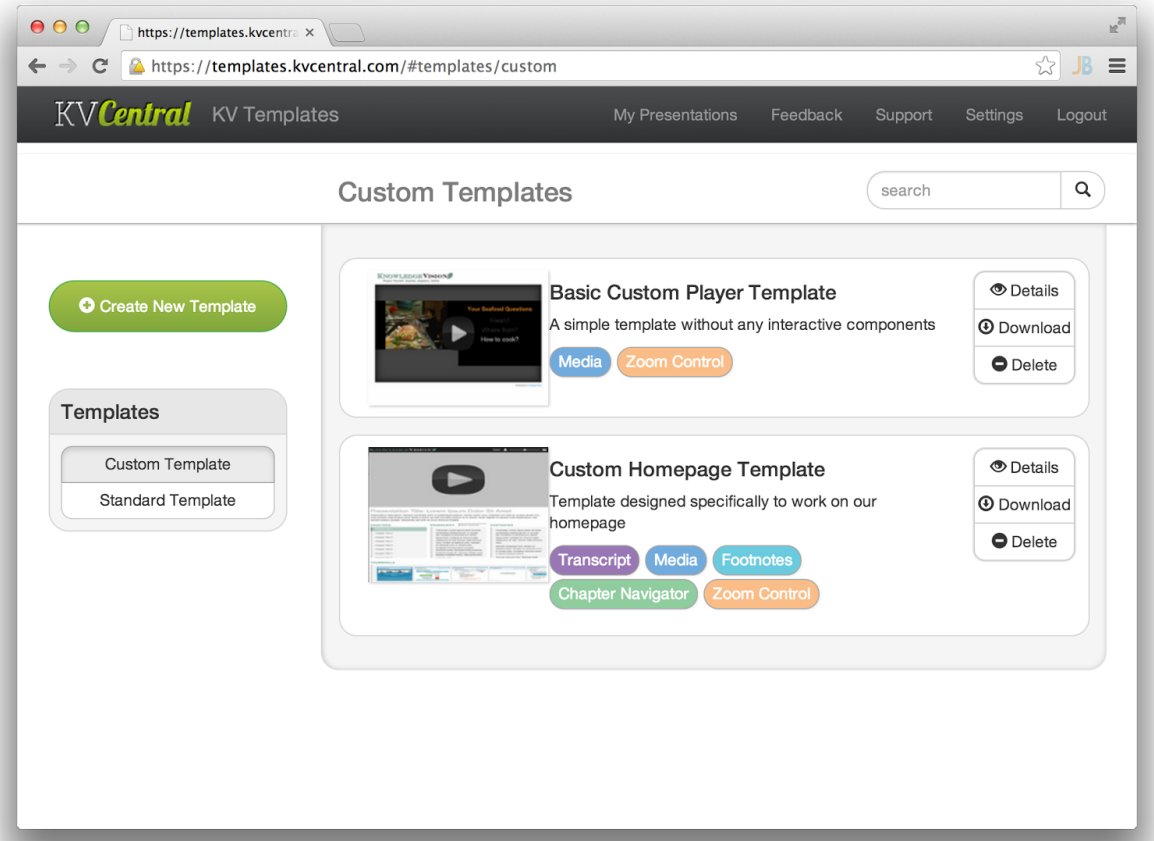

my-template-name
FAVORITES
Dropbox
Macintosh HD
All My Files
AirDrop
Desktop
akieft
Applications
Documents
Music
Pictures
Projects
Demo Materials
Shared Files
KnowledgeVision Address.txt
DEVICES
Remote Disc
T...late Example
acme-template-1
acme-template-2
acme-template-3
acme-template-4
acme-template-5
my-template-name
.DS_Store
css
img
index.html
js
Open in New Tab
Move to Trash
Get Info
Compress "my-template-name"
Burn "my-template-name" to Disc...
Duplicate
Make Alias
Quick Look "my-template-name"
Share
Paste Item
Show View Options

Uploading and Managing Templates

The Template Management Module is a web interface for uploading and managing Enhanced Templates in your KnowledgeVision account. Once you have logged into KVCentral, you can access the Template Management Module at https://templates.kvcentral.com/.

Browsing Templates

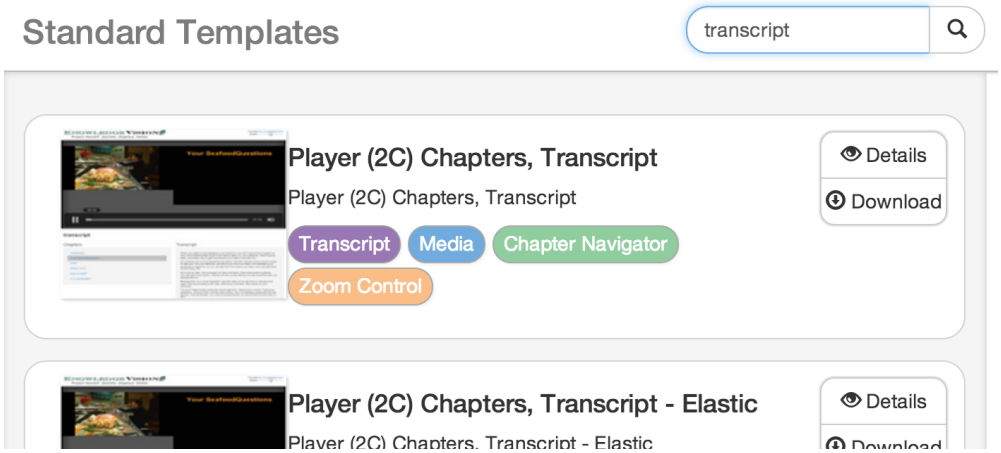
When you open the Template Management Module, you will see a list of all of the Custom Templates that have been uploaded to your account. The list may be empty if no one has created any templates in your account yet.



You can toggle between the list of **Custom Templates** and the list of **Standard Templates** by selecting between the two options. Standard Templates are available to all accounts, and therefore cannot be edited or deleted. If you would like to create a customized version of a Standard Template, you can download the ZIP file, edit it, and upload it to your account as a new Custom Template.

Each item in the list shows the name, description, thumbnail, and list of components for the template, along with buttons to view **Details** about the template and **Download** the original ZIP file. Custom Templates also provide the option to **Delete** them from your account.

You can filter either list of templates by entering a query in the **Search** field at the top. The list will refresh to show only templates that contain your query term in their name or description.

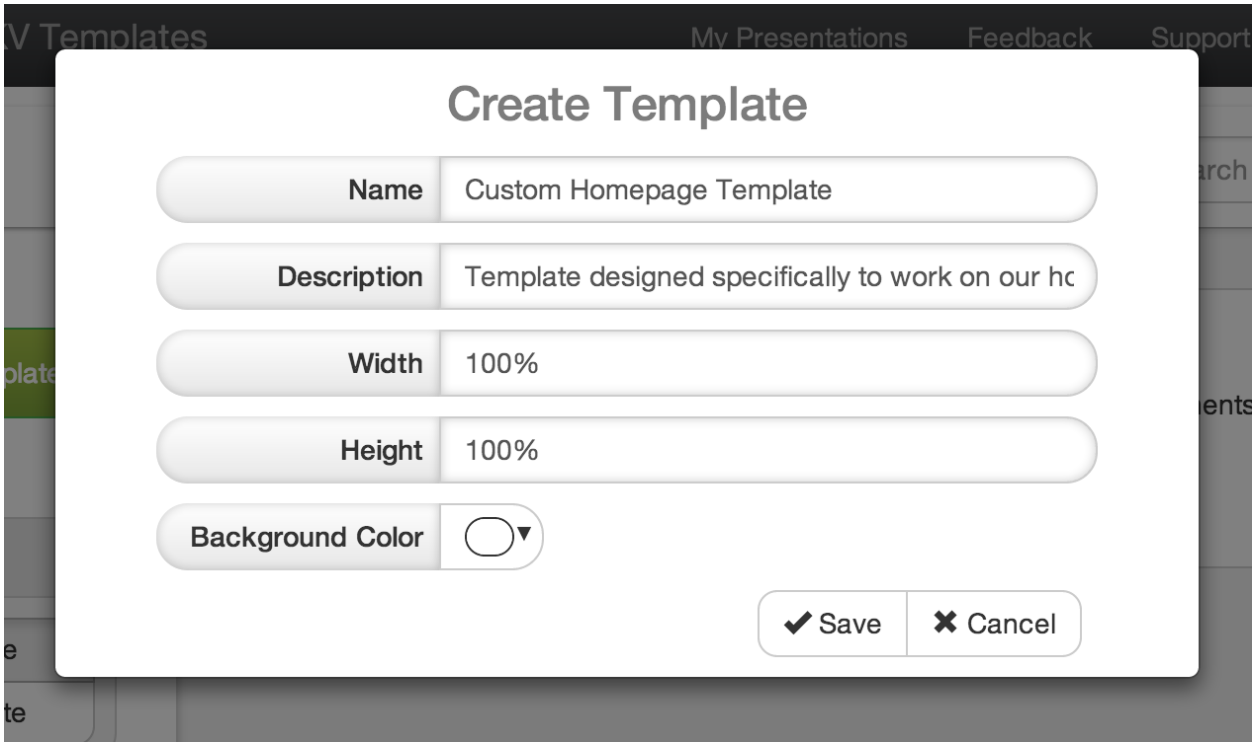


To return to the full list of templates, simply clear the search query.

Creating a New Template

To create a new Enhanced Template, click the **Create New Template** button that appears when browsing your list of Custom Templates.

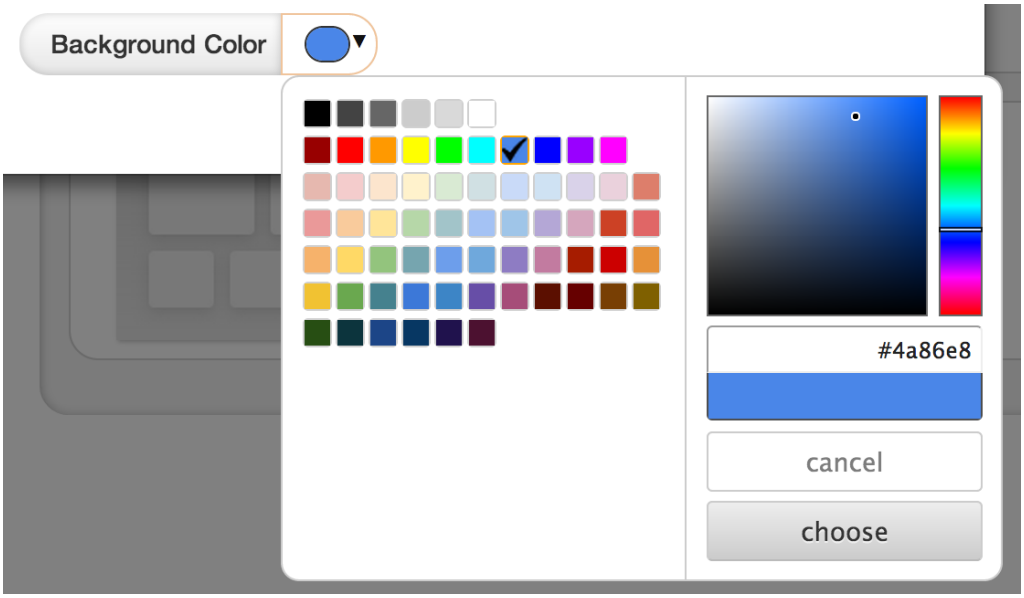
The **Create Template** window will open, where you can enter a Name, Description, Width, Height, and Background Color for your template.



The **Name** and **Description** that you enter will appear in KVStudio when users browse for your template, so it is helpful to include enough information that it will make your template easy to find.

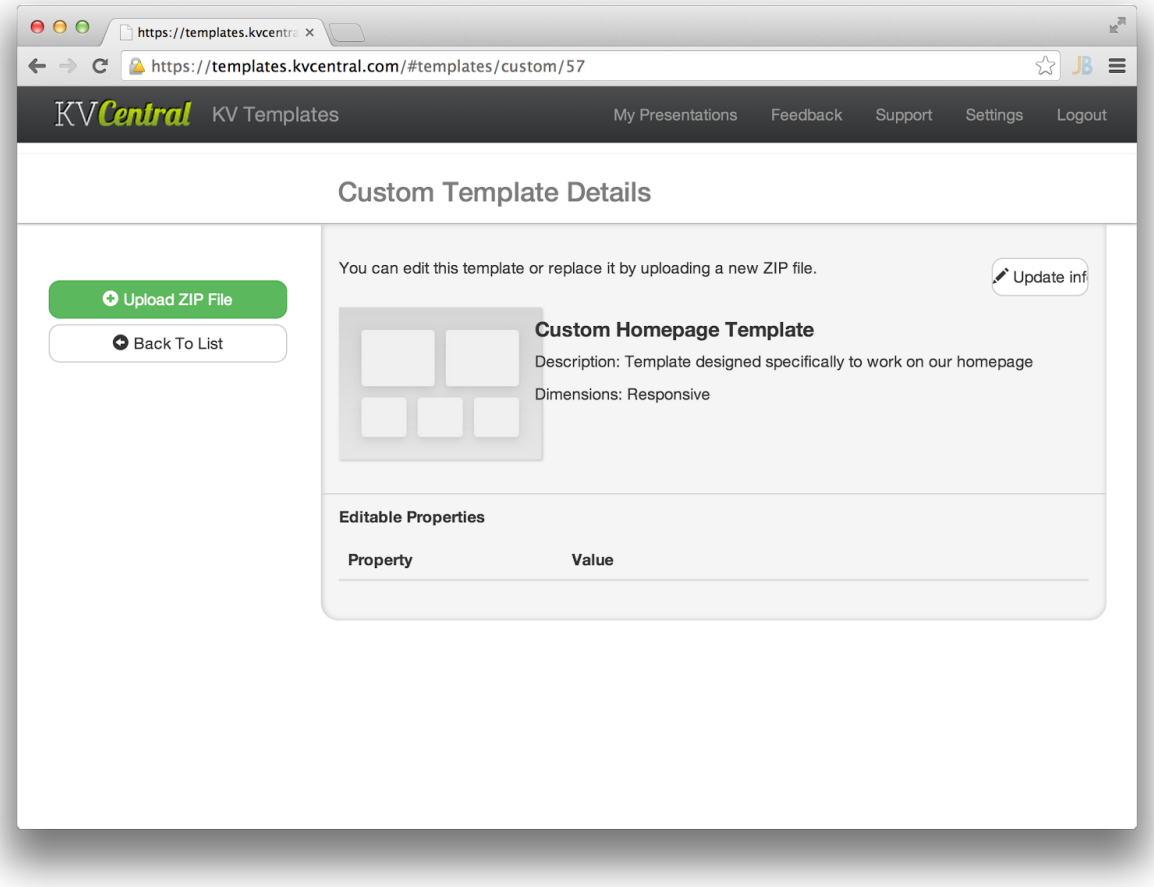
The **Width** and **Height** fields require that you enter valid CSS dimension values, either in pixels or percentages. If you would like to assign fixed dimensions to your template, enter pixel values such as `640px` or `960px`. If instead you would prefer that your template dynamically fill the available space wherever it is embedded, enter `100%` for both Width and Height. The dimensions that you enter will appear in KVStudio when users browse for your template, but if you have entered `100%`, KVStudio will display the dimensions as “Responsive.”

The **Background Color** you select will appear when a presentation published with your template is viewed directly from the `watch.knowledgevision.com` link that is generated for it. For example, if you assign fixed dimensions to your template, but the presentation is loaded in a browser window larger than those dimensions, this background color will appear in the space surrounding the template.

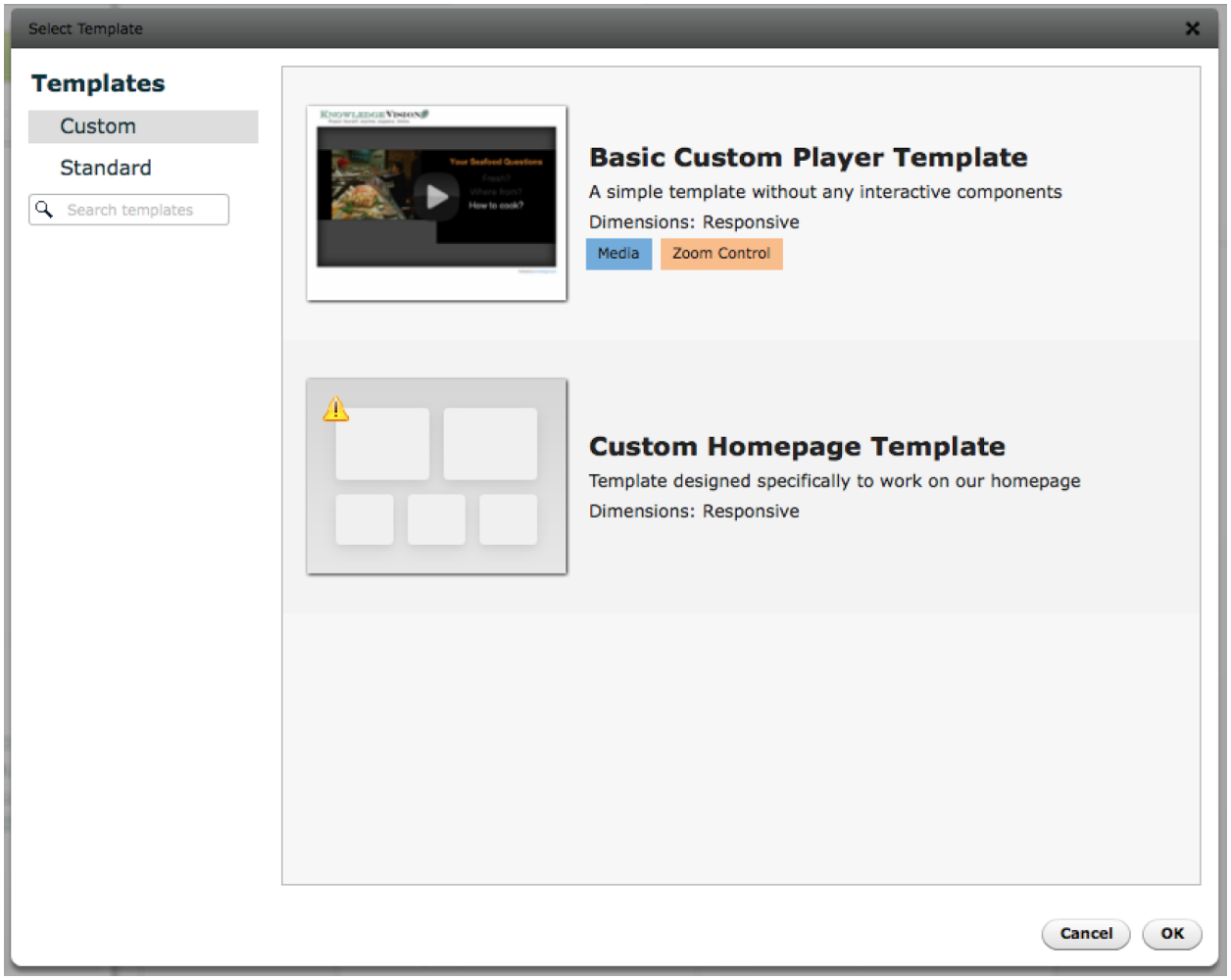


Once you have entered the appropriate information in the fields, click the **Save** button to create the template.

At this point in the process, you will have created a placeholder entry for your template. The Template Management Module will bring you to the Custom Template Details page for your new template.

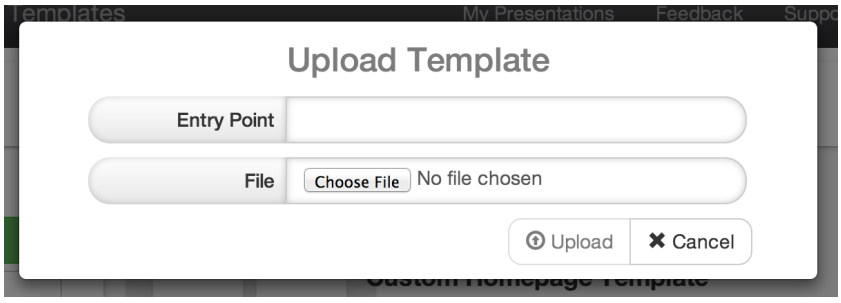


Note that until you actually upload your template ZIP file, KVStudio users will see the new template in the list, but they will not be able to select it for use in a project.



Uploading a Template ZIP File

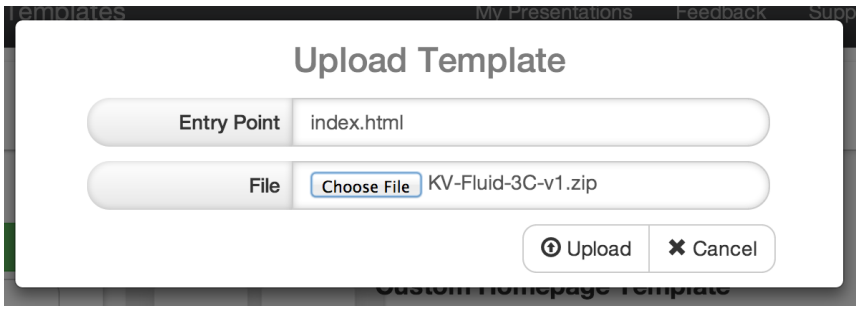
To upload your template ZIP file, click the **Upload ZIP File** button. The **Upload Template** prompt will appear.



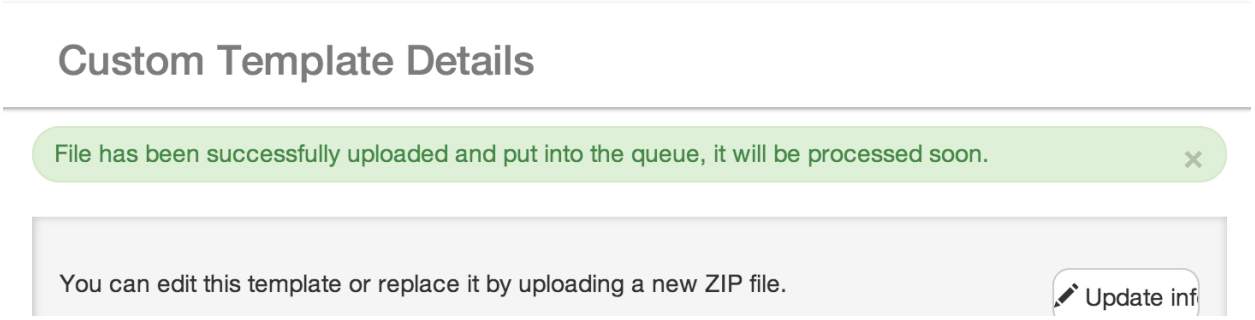
In the **Entry Point** field, you may optionally enter the filename of the primary HTML file in your template package. This is typically named `index.html`, but you may have given it a different

name. If your ZIP file only contains a single HTML file in the root folder, you can leave this field blank and the system will automatically detect it.

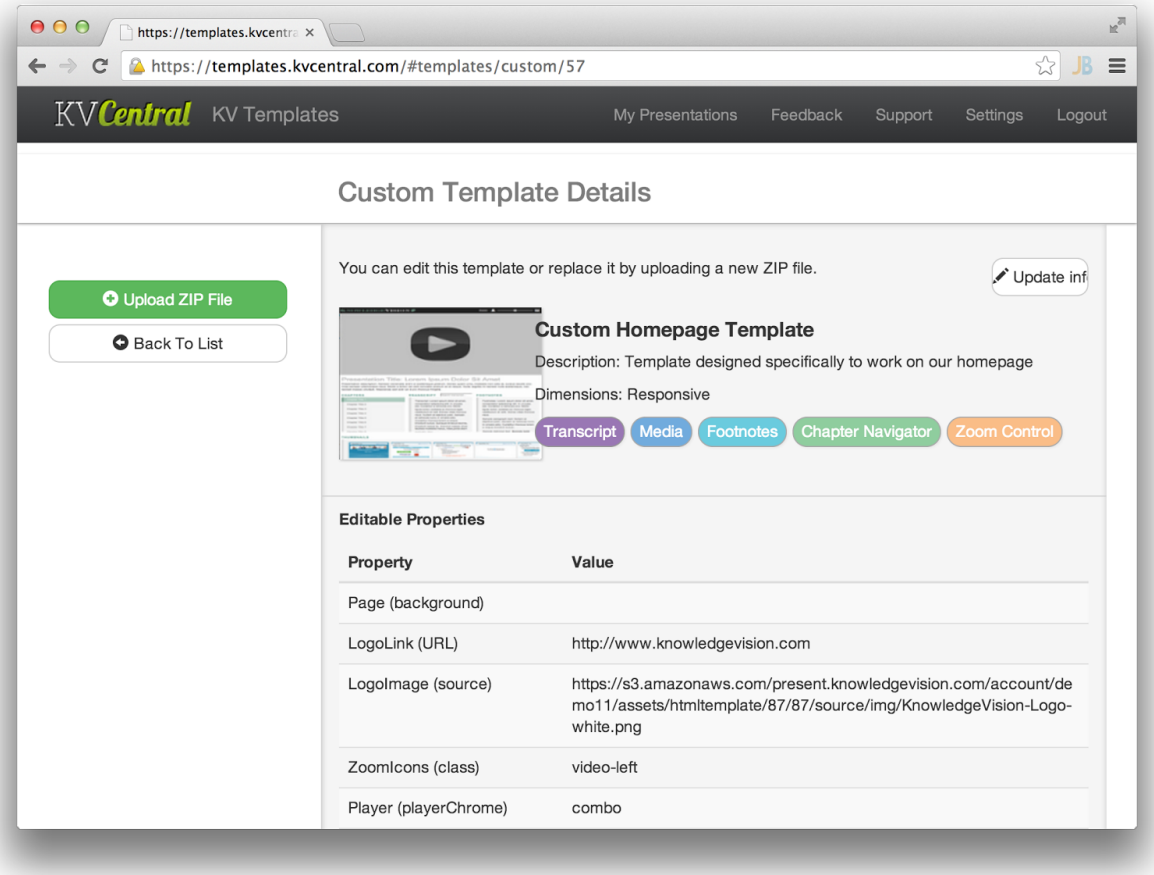
In the **File** field, click the **Choose File** button to select the .zip file from your computer's file system.



Once your file has been selected, click the **Upload** button to start the upload. If the upload completes successfully, you will see a notification informing you know that your template file is being processed by the system.



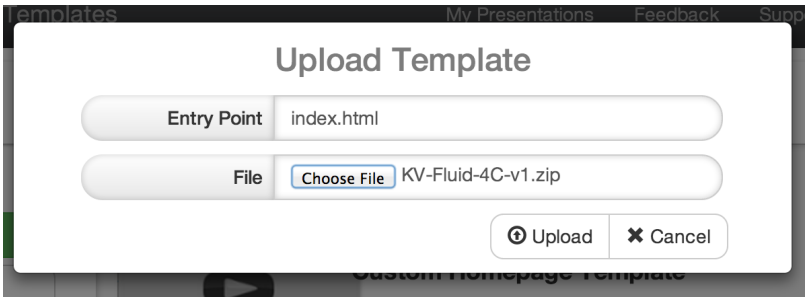
After a moment or so, you can refresh the browser page to see if the template processing has completed. Unless there was a problem, the Custom Template Details page will now display the full information about the template, including its thumbnail image, list of components, and list of Editable Properties.



Your Enhanced Template is now available for users to select in KVStudio for their projects.

Replacing a Template

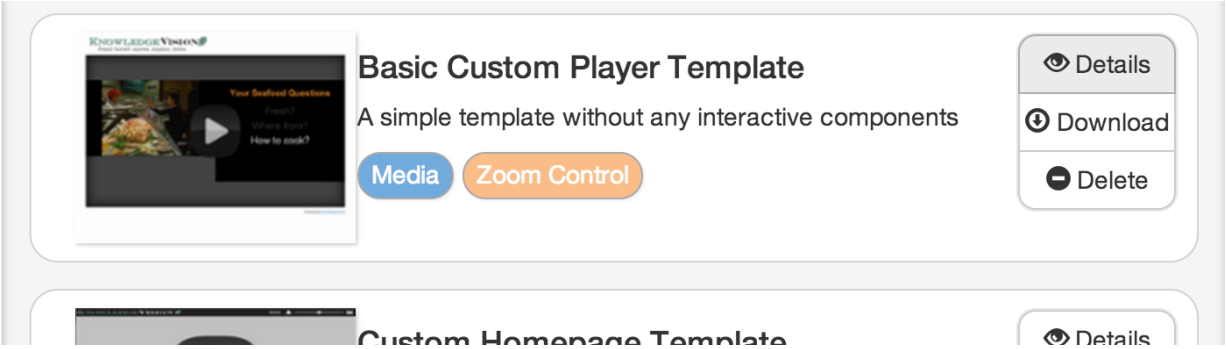
If you make any changes to your Enhanced Template and wish to replace the version you have uploaded to your account, you can always go back to the Custom Template Details page for the template and click the the **Upload ZIP File** again. You will again be prompted to enter an **Entry Point** filename and select a .zip file from your computer. The new version will now be used in place of the old one whenever a user selects the template in KVStudio.



Important Note: Currently, replacing an Enhanced Template has no effect on published presentations, which will continue to use the older version of the template. If you wish to apply your new template to published presentations, you will need to open the presentation project file, reselect the template, then republish the presentation. The presentation should now load your updated version of the template.

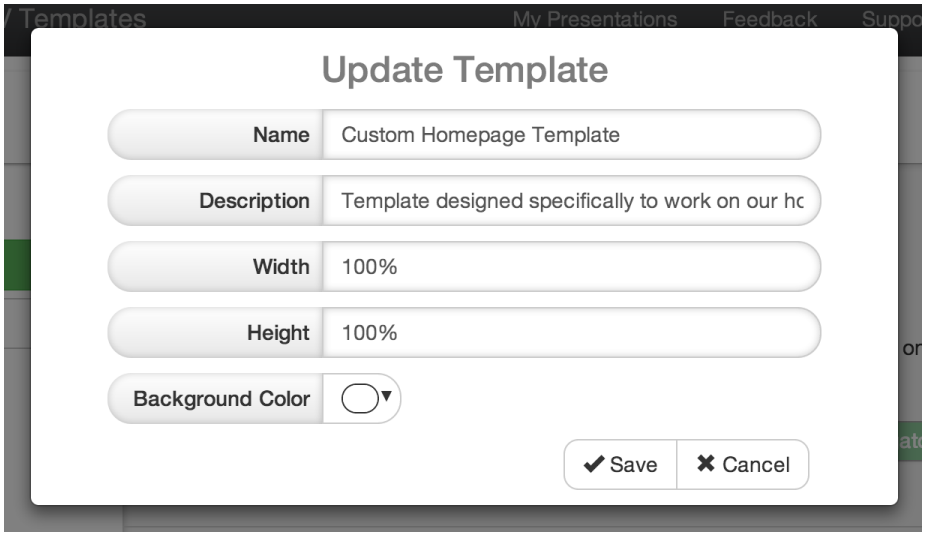
Updating Template Details

In addition to replacing the template ZIP file, you can also update the name, description, dimensions, and background color of your Enhanced Templates at any time. From your list of Custom Templates, find the template you want and click the **Details** button.



You will now see the Custom Template Details page for your template. Click the **Update Info** button.

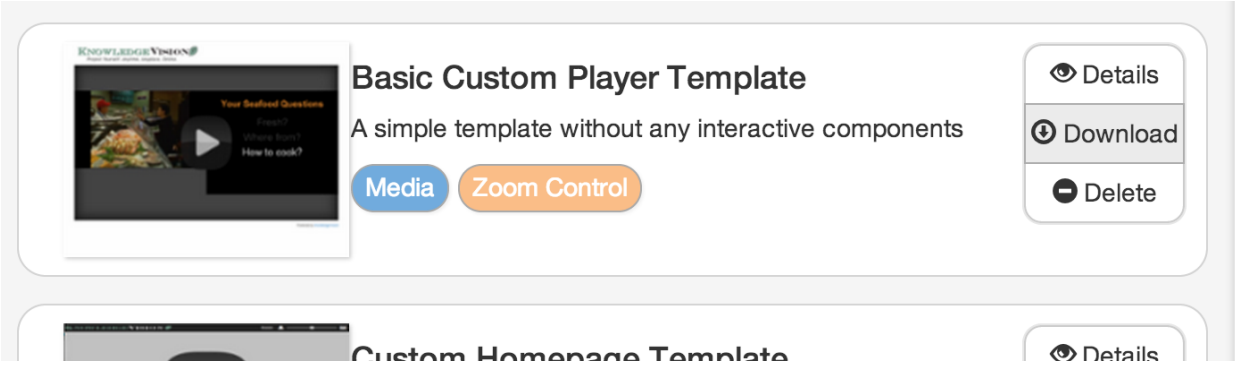
The **Update Template** window will open, allowing you to change the template information.



Click the **Save** button to save your changes. If you have changed the dimensions of a template, you may need to republish existing presentations for the change to take effect.

Downloading a Template

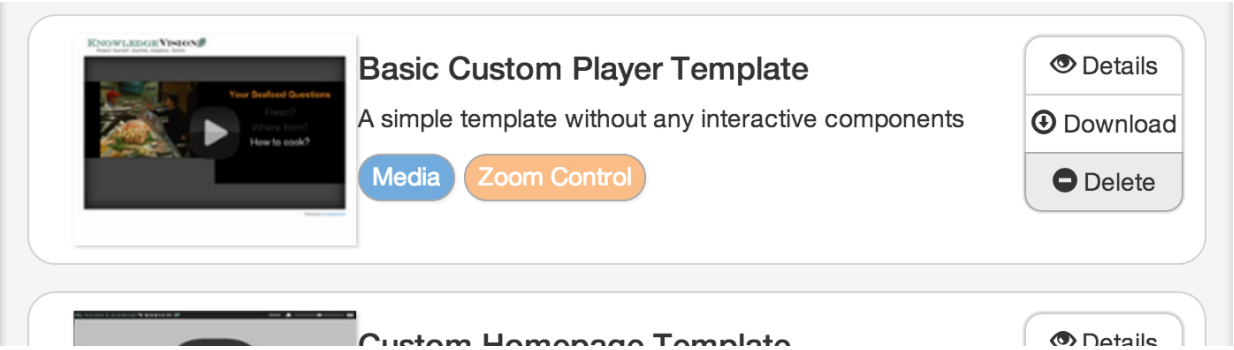
If you want to modify an existing Custom Template that you did not originally upload, or create a customized copy of a Standard Template, you can download the template's ZIP file. From either list of templates, find the template you want and click the **Download** button.



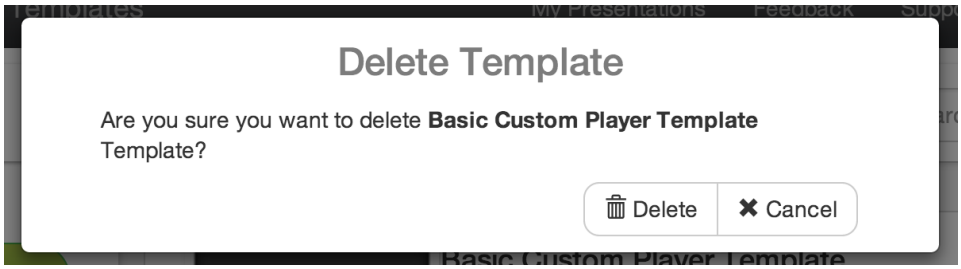
The .zip file will be downloaded to your computer file system. You can unzip the ZIP file, edit the contents, and repackage it as a new Enhanced Template.

Deleting a Template

If you would like to remove a Custom Template from your account, you can find it in the list and click the **Delete** button.



You will then be prompted to confirm that you actually want to delete the template. There is no way to restore a deleted template, though you can always upload the original ZIP file again to recreate it.



Deleting the template will remove it from your list, both in the Template Management Module and in KVStudio. However, any presentations that were previously published with the template before it was deleted will remain unaffected.

Additional Information

Please see the *KVStudio 5.0 Release Notes* for more information about using Enhanced Templates.