Adding Multimedia to Your Connexions Content[*]

Connexions
Adan Galvan
Mark Husband
Daniel Williamson

Abstract

An explanation of how to add various types of multimedia objects to your Connexions content. Examples of some of the more common multimedia objects are included.

Contents

[*]Version 1.19: Nov 12, 2010 11:56 am -0600
[†]http://creativecommons.org/licenses/by/3.0/

1 Introduction

Connexions modules can contain many different types of embedded multimedia files. This document explains how to use these elements to create media-rich modules. Though you are welcome to skip ahead at any time, we strongly recommend that you read the sections on Accessibility, Uploading Media Files, and Embedding Media Elements in Your Module before moving on to a specific medium as these sections cover several important aspects common to all media types.

> NOTE: This module contains a number of code examples designed to illustrate specific aspects of the CNXML language. As a result, these code snippets will not always take advantage of all of the available options and attributes for every element. Feel free to experiment with different combinations and structures as you build your modules to learn how these options can enhance your module's multimedia content.

Connexions supports the use of any media format that has a valid MIME type (see this page[1] for more information). In order to take advantage of the media, however, end-users must also have any necessary software and/or browser plugins installed on their browsers. For this reason, it is advisable to stick with common file formats when authoring in Connexions. This module provides a number of CNXML code snippets that cover several typical use cases, though this list is not exhaustive.

> TIP: If you need assistance adding multimedia to your modules, have questions about an object not listed in this module, or have any other authoring issues that you need help with, please contact Connexions technical support at techsupport@cnx.org[2] .

2 Making Multimedia Accessible for All Users

It is important for authors to remember that not all users are able to experience media the same way. Visually impaired users will not be able to view graphical content, while those hard of hearing will not be able to experience audio content as intended. As an author, should always strive to make their content accessible to as many audiences as possible; in addition to allowing more users to enjoy and benefit from your work, you will also make it easier for the many organizations required by policy or law to meet accessibility standards to use and adopt your content.

Alt Text

Alt text allows authors to provide a short description of the visual content of a media element. These descriptions can be read by screen reading software packages used to assist visually impaired users. Keep your alt text short but descriptive; imagine somebody trying to describe an image you cannot see and think about the information you would find most important. A few tips:

- Don't be redundant. The user is already aware that this is an image, so don't start your description with "This image is . . . " or "A picture of . . . "
- Keep it short and sweet. Describe only the relevant aspects of the image; for example, it might be important to describe the subject of a photograph as having freckles, but it is probably not necessary to count how many freckles she has on each cheek.
- Use correct grammar and spelling. For the majority of users requiring alt text descriptions, this content will be read out loud; these descriptions should adhere to the same quality and style standards used elsewhere in your work.

> IMPORTANT: Alt text is **required** for all `<media>` elements.

[1] http://www.ietf.org/rfc/rfc1341.txt?number=1341
[2] techsupport@cnx.org

longdesc

A **longdesc** (or "long description") file is used to describe images that are too complex to summarize in alt text. Rather than including the description in the CNXML markup, the full, detailed description is contained in a separate file that is linked from the `media` element. This allows you to describe the image in as much detail as necessary without needing to keep the description short. A longdesc file:

- Can be any file type, though a basic text-only HTML file is most appropriate.
- Can be loaded into a Connexions module like any other image or resource file.
- Is not a replacement for alt text, but instead is a chance for the author to elaborate on alt text.
- Is only necessary when the image is so complicated that an alt text description is not sufficient to describe it.

The following example shows how to link to a longdesc file titled `earnings_report_desc.html`:

```
<media id="longdesc_example"
       alt="An earnings graph showing steady growth of 6% over the past two quarters."
       longdesc="earnings_report_desc.html">
   <image mime-type="image/png" src="earnings_report.png" />
</media>
```

In this example, `earnings_report_desc.html` would most likely be a simple text file describing the details of the chart, including the data points, overall trends, and any other relevant information. While the alt text describes "steady growth of 6% over the past two quarters," the longdesc file can talk about specific features of the chart, such as a dip that coincided with a recent merger or a spike that represented a major product release. A screen reader user would have the option of following this link in order to get a detailed understanding of the data that would otherwise be conveyed and understood by sighted users through visual information.

TIP: Remember, longdesc files are not common, and are usually only used when the media element contains important visual information that is too complex and/or detailed for alt text.

Captioned Video and Audio Transcripts

Captioning is another important consideration when dealing with video content. Users who are hearing impaired may not be able to understand all of the information being conveyed in a video, limiting their access to the material being described. This can be addressed by submitting videos that are captioned, allowing those users to read along with the spoken materials. If you are not able to caption your videos, another option is to include audio transcripts of the materials as a module resource that can be easily accessed using a link. While there are no specific tools or standards for providing this type of content, the important thing to remember is to make it as easy as possible for as many people as possible to access the information you are trying to communicate.

NOTE: You can read more about how Connexions addresses accessibility concerns in m17212 - Accessibility Features of Connexions[3].

3 Embedding Media Elements in Your Module

3.1 The <media> Element

Once you've attached your multimedia files to your module, the next step is to embed the media files in your content. This can be done by using the `<media>` element.

The `<media>` element is the primary building block for including multimedia files in your module. In its simplest form, a `<media>` element contains three pieces of information:

[3]"Accessibility Features of Connexions" <http://cnx.org/content/m17212/latest/>

- A unique id
- Alt text
- A media subtype element, such as <image>

The code snippet below shows a simple example of <media> element:

```
<media id="media_example" alt="a dog on a bed">
   <image mime-type="image/jpeg" src="image1.jpg"/>
</media>
```

This code results in the following display:

Note that the resulting image displays **inline**; when included as the child of another element, such as <para>, the media element will display in line with the surrounding text by default:

```
<para id="myparagraph">
 <media id="media_example2" alt="a dog on a bed">
    <image mime-type="image/jpeg" src="image1.jpg"/>
 </media>

 This is my dog.  Isn't he cute?

</para>
```

This code results in the following display:

This is my dog. Isn't he cute?

Each <media> must contain a **media subtype** element from the following list:

- <image>
- <video>
- <flash>
- <audio>
- <java-applet>
- <labview>
- <text>
- <download>

3.2 The <figure> Element

The <figure> element allows you to set a media element apart from the surrounding text and highlight it as a labeled figure within the module. In it's simplest form, a <figure> contains two pieces of information:

- A unique ID
- A <media> element

The code snippet below shows an example of a <figure> element:

```
<figure id="figure_example">
 <media id="dog_on_bed" alt="a dog on a bed">
   <image mime-type="image/jpeg" src="image1.jpg"/>
 </media>
</figure>
```

This example results in the following:

Figure 1

You'll notice that the figure is labeled as "Figure 1" - this information was not provided in the CNXML code, but is instead supplied automatically by Connexions based on the figure's place within the document. As you add, move, and delete figures in your module, you do not have to worry about keeping track of the figure numbering - this is done for you when the page is displayed to the reader. Also, unlike the <`media`> example shown previously, the image in this example is displayed in **block** mode rather than **inline** mode.

> TIP: When referring to figures in your Connexions modules, you can take advantage of the figure's unique ID to create a **dynamic reference** that is automatically labeled for you. To do so, simply create an empty <`link`> that points to the <`figure`>; in this case,
>
> ```
> <link target-id="figure_example" />
> ```
>
> results in the following link: Figure 1.
>
> Connexions will automatically label the text of the link to match the figure's label. Dynamic references can also be created for other "numbered" elements, such as <`equation`>, <`example`>, <`exercise`>, etc.

The <`figure`> also provides additional options that allow you to further highlight and describe an enclosed <`media`> element. The following example illustrates several additional elements:

- <`title`>, which adds a title to the figure.
- <`caption`>, which adds a caption allowing the author to provide a description or context for the figure.
- <`label`>, which overrides the default 'Figure' label; this is especially useful for modules not written in English.

```
<figure id="figure_example_extended">
 <label>Figura</label>
```

```
 <title>Mi Perro Benny</title>
 <media id="dogpic" alt="Perro sentado en la cama">
  <image mime-type="image/jpeg" src="image1.jpg" />
 </media>
 <caption>Este es mi perro Benny haciendo lo que hace mejor.</caption>
</figure>
```

This example results in the following display:

Mi Perro Benny

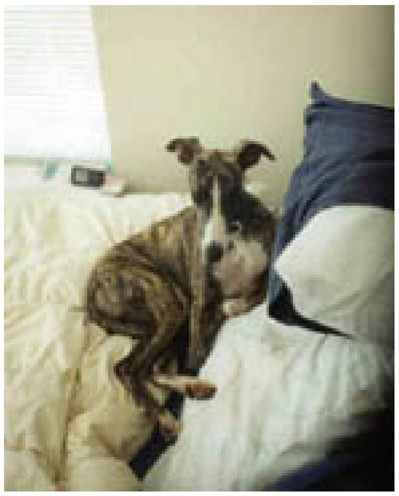


Figura 2: Este es mi perro Benny haciendo lo que hace mejor.

> TIP: Remember, the `<figure>` element is not necessary to display a `<media>` element. Use `<figure>` when you want to set your multimedia content apart from the surrounding text, add a title or caption, or create a numbered figure label for your media. If you just want your media to display inline, only use the `<media>` element.

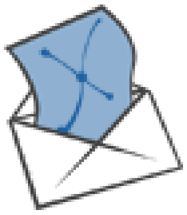
3.3 The <subfigure> Element

You can place two or more images within the same figure using the `subfigure` tag. This is useful when you have two related images that you want to display side-by-side or one above the other. The following CNXML snippet can be used to display two subfigures side-by-side:

```
 <figure id="figure-2" orient="horizontal">
<subfigure id="subfigure-1">
   <title>Blue on Blue</title>
   <media id="sub_example" alt="An envelope with a blue page">
      <image mime-type="image/png" src="Xenvelope-blue-on-blue.png"/>
      <caption>Subfigure 1.</caption>
   </media>
</subfigure>
```

```
   <subfigure id="subfigure-2">
      <title>Orange on White</title>
      <media id="sub_example2" alt="An envelope with a white page">
         <image mime-type="image/png" src="Xenvelope.png"/>
         <caption>Subfigure 2.</caption>
      </media>
   </subfigure>
   <caption>Two images displayed horizontally in one figure.</caption>
</figure>
```

The resulting figure is displayed below:


(a) Blue on Blue

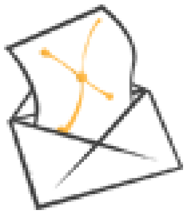
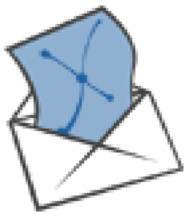

(b) Orange on White

Figure 3: Two images displayed horizontally in one figure. (a) Subfigure 1. (b) Subfigure 2.

The optional `orient` attribute determines how the subfigures are arranged respective to one another. This attribute can have a value of either `horizontal` (default) or `vertical`. Modifying the previous example so that `orient="vertical"` results in the following:


(a) Blue on Blue

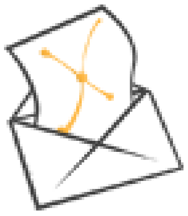

(b) Orange on White

Figure 4: Two images displayed vertically in one figure. (a) Subfigure 1. (b) Subfigure 2.

TIP: Only use subfigures when the images are related to one another.

3.4 Including Alternate Media for Print

Connexions allows you to insert an alternate media file in your module that can be formatted and sized specifically for the print version of your content. For example, you may want to provide a color image for online use and a black-and-white image for print. You can also use this feature to provide alternate media types, such as using a Java applet for online use and an image for print.

To create an alternate image for print, simply include two `image` elements (or, say, a `video` element and an `image` element) inside the `media` tag. Use the `for` attribute to specify whether this element should be used for the online version, the PDF version, or used as the default selection.

The following code gives one example for including an alternate print image. In this case, online users will see `image.png`, while `image.eps` will be used for the print version of the module.

```
<figure id='printimage'>
   <media id="print_image_example">
      <image mime-type='image/png' src='image.png'/>
      <image mime-type='application/postscript' src='image.eps' for="pdf"/>
   </media>
</figure>
```

Possible values for the `for` attribute include:

default: (Default) This element is used for both PDF and online versions unless another, version-specific element is present.
online: This element is used only for the online version of the content.
pdf: This element is used only for the pdf version of the content.

Note that it is not necessary to specify a media element for both online and print versions. For example, given the code snippet above, the first image ("`image.png`") is the default and will be automatically used for the online version of the content, while the second image ("`image.eps`") is explicitly assigned to the PDF version of the content. If no default had been specified, then no image would have appeared in the online version of the content.

TIP: You can use any two media objects in this way to create alternative media for the online and PDF versions of your content. When two media subtype objects are included inside a `media` element, use the `for` attribute to specify when that object should be displayed.

4 Media Subtypes

Within each <`media`> element, at least one media subtype is required. The following sections describe how to create the various possible subtype elements.

4.1 Images

Some of the image file types and their corresponding MIME types used in Connexions modules are:

eps - mime-type="application/postscript"
png - mime-type="image/png"
jpg - mime-type="image/jpeg"

gif - mime-type="image/gif"

This list is not all inclusive. Any image with a valid MIME type[4] can be used in a module.

4.1.1 Embedding Images

To embed images in your module, insert CNXML entries similar to the following example into the "index.cnxml" file for your module:

```
<figure id="figure-01">
   <title>Example Figure</title>
   <media id="figureexample" alt="an envelope with a blue page">
      <image mime-type="image/png" src="Xenveope-blue-on-blue.png"/>
   </media>
   <caption>A graphic image displayed as a figure.</caption>
</figure>
```

The `src` attribute in the image tag gives the location or source of the image that you want to appear in the module. After loading the image file into the 'Files' tab on your module, simply provide the full file name (e.g. `"envelope-taller.png"`) to specify which image should be used.

The actual display (Figure 5: Example Figure) for the previous CNXML example is:

Example Figure

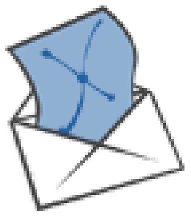


Figure 5: A graphic image displayed as a figure.

4.1.2 Adjusting the Size of the Image

You can use optional size attributes to control the size of the image displayed in both online and print versions of the content. There are a few things to keep in mind when specifying image sizes in CNXML:

- Images that are displayed larger than their original size will look fuzzy or grainy.
- Resizing an image online relies on the web browser to determine how to redraw the image. This can sometimes lead to unpredictable and undesirable results that vary from browser to browser. Whenever possible, resize the image to the desired dimensions before loading the file into Connexions to avoid this problem.
- It is not necessary to specify both the height and width of an image. If you supply only one of these, the other value will be calculated to preserve the same height-width ratio as the original image. You only need to supply both when you wish to change the shape of the image.

[4]http://www.duke.edu/websrv/file-extensions.html

- You can choose to resize an image for online viewing, printing, or both by supplying the corresponding attributes as described below. Sizes specified for one version have no effect on the other. If you do not supply sizing information, the image will be displayed in its original size by default.

4.1.2.1 Adjusting the Size of an Online Image

You can adjust the size of the image in the on-line version of your document by using the `height` and `width` attributes.

The following code will display the image in its original, default size:

```
 <figure id="element-439">
   <media id="tajmahal" alt="The Taj Mahal">
     <image src="TajMahal_medium.jpg" mime-type="image/jpeg"/>
   </media>
   <caption>
     The photograph of the Taj Mahal in this figure and the following figures
     was taken by <link url="http://www.flickr.com/photos/babasteve/">Steve
     Evans</link>.  It is licensed for public use under the Creative Commons
     Attribution License.
   </caption>
</figure>
```

This code results in the following image:

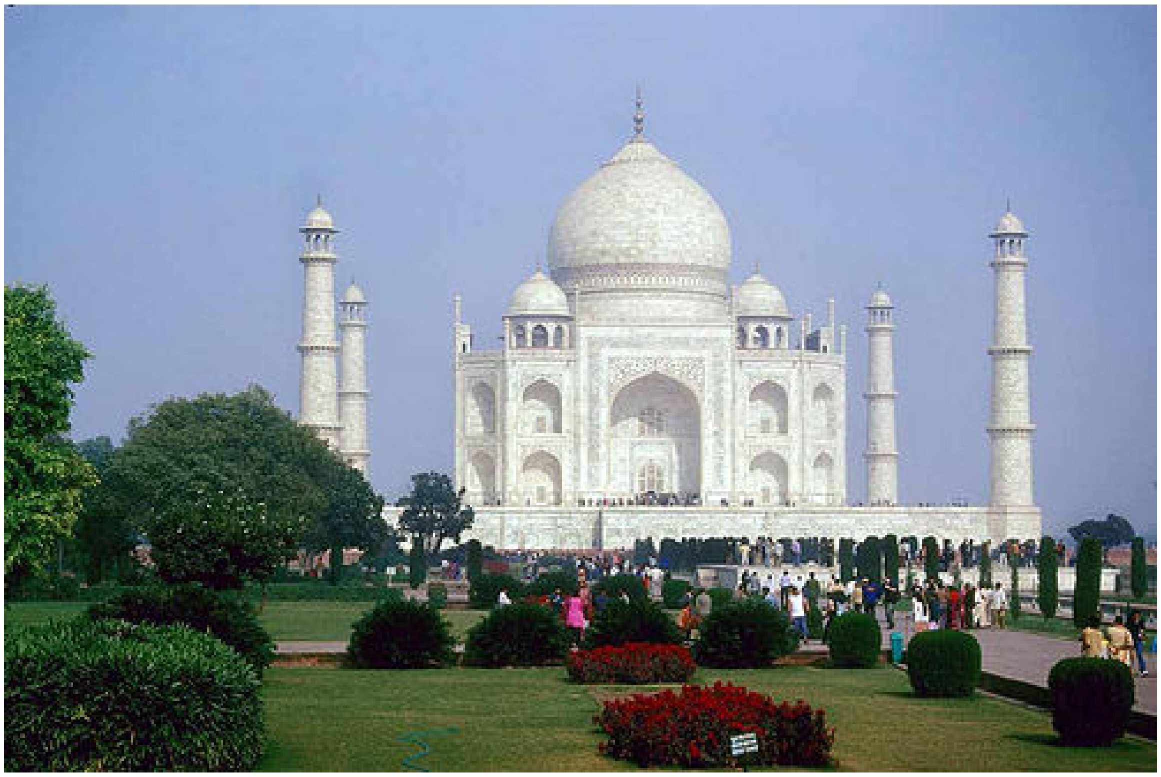


Figure 6: The photograph of the Taj Mahal in this figure and the following figures was taken by Steve Evans[5]. It is licensed for public use under the Creative Commons Attribution License.

The following CNXML code shows how to display the same image with a specific size (in this case, 250 pixels wide by 250 pixels tall):

```
   <figure id="figure_size_adjusted">
  <media id="TajMahal" alt="The Taj Mahal">
     <image mime-type="image/jpeg"
            src="TajMahal_medium.jpg"
            height="250"
            width="250" />
  </media>
</figure>
```

The resulting figure is displayed below:

[5]http://www.flickr.com/photos/babasteve/

Figure 7

When specifying `width` and `height`, enter the value in pixels. For example, the code above will produce a print image that is 250 pixels tall and 250 pixels inches wide.

> TIP: It is not necessary to specify both the `height` and `width` of an image. If you supply only one of these, the other value will be calculated to preserve the same height-width ratio as the original image. You only need to supply both when you wish to change the shape of the image.

> TIP: The `height` and `width` attributes do not affect the size of the image in the print (PDF) version of the module. To change the printed size of the image, use the `print-width` attribute (see below). You can use any combination of print and online sizing attributes for an image.

4.1.2.2 Adjusting the Size of a Printed Image

You can adjust the size of the image in the print (PDF) version of your document by using the `print-width` attribute. The height of the image will be adjusted proportionately to the width set in `print-width`. There is no `print-height` attribute.

The following CNXML code shows how to display the Taj Majal image above with a specific printed size (in this case, 4.5 inches wide):

```
 <figure id="figure_size_adjusted">
<media id="TajMahal" alt="The Taj Mahal">
  <image mime-type="image/jpeg"
       src="TajMahal_medium.jpg"
```

```
             print-width="4.5in" />
   </media>
</figure>
```

When specifying `print-width` include the unit of measurement in the value. This can be `in` (inches), `cm` (centimeters), `pt` (points) or any other value supported in LaTeX[6] . The code above will produce a print image that is 4.5 inches wide and a height proportional to its original dimensions.

> TIP: The `print-width` attribute does not affect the size of the image when viewing the version of the module. To change the online size of the image, use the `height` and `width` attributes. You can use any combination of print and online sizing attributes for an image.

4.1.2.3 Creating a Linked Thumbnail Image

A thumbnail is a small version of an image that, when clicked, displays the larger, full-size version of that image. You can create a thumbnail in your modules by completing the following steps:

1. Using image-editing software, create a thumbnail-sized version of your image file. Make sure to change the name of the thumbnail image so that you know which is which (e.g. "myimage_thumb.png").
2. Add the thumbnail image to your module using the files tab as you would with any image fie.
3. Create the `media` and `image` elements as you would normally do for a full-sized image.
4. Use the optional `thumbnail` attribute to the `image` element as shown in the example below.

Here is an example of the CNXML code required to include a thumbnail image that links to a full size version:

```
    <figure id='thumbnail'>
   <media id="thumbnailmedia" alt="The Taj Mahal (thumbnail version)">
      <image mime-type='image/jpeg'
             src='TajMahal_medium.jpg'
             thumbnail='TajMahal_Thumb.jpg'/>
      </image>
   </media>
</figure>
```

This code results in the following linked image:

[6]http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/teTeX/latex/latex2e-html/ltx-86.html

Figure 8

4.2 Video

4.2.1 Embedding Videos Attached to the Module

Movies can be added to your module using the same basic structure as with other media types.

```
    <figure id='moviefig2'>
   <media id="movie" alt="Building on the Past">
      <video mime-type="video/mpeg"
             src="Building_on_the_Past.mpg"
             width="350"
             height="300"
             autoplay="false" />
   </media>
   <caption>
     The Creative Commons movie: "Building on the Past".
     Click the Play button to start the movie.
   </caption>
</figure>
```

The movie is displayed below:

This media object is a video file. Please view or download it at
<http://cnx.org/content/m12660/1.19/Building_on_the_Past.mpg>

Figure 9: The Creative Commons movie: "Building on the Past". Click the Play button to start the movie.

4.3 Flash Objects

You can insert Flash objects in your module. Here is an example of the CNXML code required to include a Flash object:

```
    <figure id='flashfig'>
   <media id="new_flash" alt="flash animation of the connexions logo">
      <flash mime-type="application/x-shockwave-flash"
             src="flash.swf"
             height="250"
             width="250" />
      </flash>
   </media>
   <caption>"Welcome to Connexions" example Flash object.</caption>
</figure>
```

The Flash object (Figure 10) appears in your module the same way in which a movie or an image appears:

This media object is a Flash object. Please view or download it at
<http://cnx.org/content/m12660/1.19/flash2.swf>

Figure 10: "Welcome to Connexions" example Flash object.

4.4 Audio Files

Audio files such as mp3, real audio, and wav files can be inserted into your module quickly and easily. To include audio files in your document, upload the corresponding audio file and include code similar to the following in your "index.cnxml" document:

```
<figure id='musicscale'>
   <media id="music_example" alt="chromatic scale slurred">
      <audio mime-type="audio/x-wav" src="chromatic_slurred.wav"/>
   </media>
   <caption>A chromatic scale performed on clarinet by Michael Lawrence.</caption>
</figure>
```

The actual display (Figure 11) for the previous CNXML example is:

This media object is an audio file. Please view or download it at
<http://cnx.org/content/m12660/1.19/chromatic_slurred.wav>

Figure 11: A chromatic scale performed on clarinet by Michael Lawrence.

4.5 Java Applets

Java applets are another media type you can use in your Connexions modules:

```
<figure id="javafig">
   <media id="java_example" alt="a complex sinusoid">
      <java-applet mime-type="application/x-java-applet"
                   code="PhasorDemo.class"
                   width="430"
                   height="500" />
   </media>
   <caption>3D Animation of a Complex Sinusoid. Click the Run button to start the animation.</caption>
</figure>
```

The actual display (Figure 12) for the previous CNXML example is:

This media object is a Java applet. Please view or download it at
<http://cnx.org/content/m12660/1.19/#java_example>

Figure 12: 3D Animation of a Complex Sinusoid. Click the Run button to start the animation.

If the graphic does not appear in the figure above, you may need to install or upgrade the Java plug-in on your computer. Please go to java.com[7] to download the latest version of the Java plug-in.

4.6 LabVIEW Demonstrations

For information on including LabVIEW demonstrations in your module, please see Creating LabVIEW demonstrations for Connexions[8].

4.7 Parameters

Some multimedia objects need options or parameters to display properly. You can pass this information to the multimedia objects with the `param` tag. The `param` tag allows you to specify the run-time settings for an object inserted into XHTML documents.

Here is an example of a `param` tag for an audio (wav) object:

```
    <figure id="audio2">
  <media id="music_example2" alt="Slurred chromatic scale">
    <audio mime-type="audio/x-wav" src="chromatic_slurred.wav">
```

[7]http://java.com
[8]"Creating LabVIEW demonstrations for Connexions" <http://cnx.org/content/m11601/latest/>

```
      <param name="title" value="Chromatic Scale"/>
    </audio>
  </media>
</figure>
```

The actual display (Figure 13) for the previous example is

This media object is an audio file. Please view or download it at <http://cnx.org/content/m12660/1.19/chromatic_slurred.wav>

Figure 13

5 Using Third-Party Services

It is possible to embed videos and other multimedia files from sites such as YouTube[9] or SlideShare[10] into your module, just as you might embed them on your own personal web site. Please keep in mind, however, that unlike media files attached to Connexions modules, there is no guarantee that these resources will be always be available to your readers. Also, content hosted on third-party websites is not necessarily released under the same open licenses as Connexions content, meaning that other authors won't have the same rights to edit and modify content as they would with an attached media file. For these reasons we strongly encourage you to consider providing your own attached media files whenever possible.

Most resources hosted by a third-party service can be embedded using the same process as an attached video or flash file as described earlier, with the `src` attribute pointing to the URL of the video rather than a local file. Some of these external resources, however, require a little extra work by the author.

5.1 Embedding YouTube Videos

If you choose to include YouTube videos in your module, you will need to first copy the information provided in the **Embed** HTML snippet provided by YouTube for the video you wish to use. As an example, consider this video[11] of Neil Armstrong walking on the moon. The Embed snippet for this video is as follows:

```
<object width="425" height="344">
 <param name="movie" value="http://www.youtube.com/v/RMINSD7MmT4&hl=en&fs=1&rel=0"></param>
 <param name="allowFullScreen" value="true"></param>
 <param name="allowscriptaccess" value="always"></param>
 <embed src="http://www.youtube.com/v/RMINSD7MmT4&hl=en&fs=1&rel=0"
        type="application/x-shockwave-flash" allowscriptaccess="always"
        allowfullscreen="true" width="425" height="344">
 </embed>
</object>
```

Since this code is designed for use on websites and not the CNXML language, the majority of this snippet is not necessary. The part that **is** important is the **embed tag**:

[9]http://www.youtube.com/
[10]http://www.slideshare.net
[11]http://www.youtube.com/watch?v=RMINSD7MmT4

```
<embed src="http://www.youtube.com/v/RMINSD7MmT4&hl=en&fs=1&rel=0"
       type="application/x-shockwave-flash" allowscriptaccess="always"
       allowfullscreen="true" width="425" height="344">
```

The CNXML example below illustrates how to embed a YouTube video in your module. Notice that it is implemented like a standalone video, with a few key features:

- The media subtype is `<flash>`, not `<video>` - this is because the YouTube player is itself a flash object. In practice, the `<video>` element would work just fine to embed a simple YouTube player, but some features (such as being able to view the movie in full screen mode) would not be available.
- The **mime-type** attribute is set to "application/x-shockwave-flash" - this is the format used by YouTube videos.
- The **src** attribute is set to the **embed** URL for the video, **not the URL for the video on the YouTube website**. The embed URL is the one found in the code snippet as shown above and is always of the form `http://www.youtube.com/v/[VideoID]` (where `[VideoID]` is the ID of the video you wish to include in your module).
- After you copy the embed URL, you need to replace the `&` characters with `&amp;`. The `&` character is a special character in CNXML and must be escaped in this manner, otherwise you will get an error when you try to save your module. In this example,

  ```
  src="http://www.youtube.com/v/RMINSD7MmT4&hl=en&fs=1&rel=0"
  ```

  becomes

  ```
  src="http://www.youtube.com/v/RMINSD7MmT4&amp;hl=en&amp;fs=1&amp;rel=0"
  ```

The final CNXML code for embedding this movie is as follows:

```
<figure id="moonlanding-youtube">
 <title>One Small Step for (a) Man</title>
 <media id="yt-media" display="block" alt="Video of the Neil Armstrong on the Moon">
  <flash id="yt-video"
        mime-type="application/x-shockwave-flash"
        src="http://www.youtube.com/v/RMINSD7MmT4&amp;hl=en&amp;fs=1&amp;rel=0"
        width="425"
        height="344">
   <param name="allowscriptaccess" value="always" />
   <param name="allowsfullscreen" value="true" />
  </flash>
 </media>
</figure>
```

And results in the following:

One Small Step for (a) Man

This media object is a Flash object. Please view or download it at
<http://www.youtube.com/v/RMINSD7MmT4&amp;hl=en&amp;fs=1&amp;rel=0>

Figure 14

TIP: Though the steps described above may look intimidating, the actual process isn't actually that hard. To embed your YouTube video, just perform the following steps:

Step 1. Copy the example code above into your module.
Step 2. Replace the URL with the one provided in the <embed> element in the HTML snippet provided by YouTube.
Step 3. Escape the ampersand characters in the URL by replacing `&` with `&amp;`.
Step 4. Adjust the height and width parameters to change the size of the video (optional).

If you are including more than one YouTube video and are using the code provided, don't forget to change the `id` for the `<figure>`, `<media>`, and `<flash>` elements each time you add a new video.

5.2 Embedding SlideShare Presentations

If you choose to include SlideShare presentations in your module, you will need to first copy the information provided in the **Embed** HTML snippet provided by SlideShare for the presentation you wish to use. As an example, consider this slideshow[12] taken from our standard Connexions presentation. The Embed snippet for this presentation is as follows: **Connexions: Create Globally, Educate Locally**

```
    <object style="margin:0px" width="425" height="355">
 <param name="movie"
  value="http://static.slidesharecdn.com/swf/ssplayer2.swf?doc=cnxdemo-090820214427-phpapp02&stripped_ti
 <param name="allowFullScreen" value="true"/>
 <param name="allowScriptAccess" value="always"/>
 <embed
  src="http://static.slidesharecdn.com/swf/ssplayer2.swf?doc=cnxdemo-090820214427-phpapp02&stripped_titl
  type="application/x-shockwave-flash" allowscriptaccess="always" allowfullscreen="true"
  width="425" height="355">
 </embed>
</object>
```

Since this code is designed for use on websites and not the CNXML language, the majority of this snippet is not necessary. The part that **is** important is the **embed tag**:

```
 <embed
  src="http://static.slidesharecdn.com/swf/ssplayer2.swf?doc=cnxdemo-090820214427-phpapp02&stripped_titl
  type="application/x-shockwave-flash" allowscriptaccess="always" allowfullscreen="true"
  width="425" height="355">
 </embed>
```

The CNXML example below illustrates how to embed a SlideShare player in your module. Notice that it is implemented like a standalone flash object, with a few key features:

- The **src** attribute is set to the **embed** URL for the slideshow, **not the URL for the page on the SlideShare website**. The embed URL is the one found in the code snippet as shown above and is of the form `"http://static.slidesharecdn.com/swf/ssplayer2.swf?doc=[PresentationID]"` (where `[PresentationID]` is the ID and title of the slideshow you wish to include in your module).
- After you copy the embed URL, you need to replace the `&` characters with `&amp;`. The `&` character is a special character in CNXML and must be escaped in this manner, otherwise you will get an error when you try to save your module. In this example,

[12] http://www.slideshare.net/cnxorg/cnxdemo

```
    src="http://static.slidesharecdn.com/swf/ssplayer2.swf?doc=cnxdemo-090820214427-phpapp02&
stripped_title=cnxdemo"
```

becomes

```
    src="http://static.slidesharecdn.com/swf/ssplayer2.swf?doc=cnxdemo-090820214427-phpapp02&amp;
stripped_title=cnxdemo"
```

The final CNXML code for embedding this presentation is as follows:

```
<figure id="slidesharefigure">
 <media id="slidesharemedia" alt="Slide show introducing the ideas behind Connexions.">
  <flash id="slideshareflash" height="355" width="425" mime-type="application/x-shockwave-flash"
   src="http://static.slidesharecdn.com/swf/ssplayer2.swf?doc=cnxdemo-090820214427-phpapp02&amp;strippe
   <param name="allowScriptAccess" value="always"/>
   <param name="allowFullScreen" value="true"/>
  </flash>
 </media>
</figure>
```

And results in the following:

Connexions: Create Globally, Educate Locally

This media object is a Flash object. Please view or download it at <http://static.slidesharecdn.com/swf/ssplayer2.swf?doc=cnxdemo-090820214427-phpapp02&stripped_title=cnxdemo>

Figure 15

TIP: Though the steps described above may look intimidating, the actual process isn't actually that hard. To embed your SlideShare presentation, just perform the following steps:

Step 1.Copy the example code above into your module.
Step 2.Replace the URL with the one provided in the <embed> element in the HTML snippet provided by SlideShare.
Step 3.Escape the ampersand characters in the URL by replacing `&` with `&amp;`.
Step 4.Adjust the height and width parameters to change the size of the presentation (optional).

If you are including more than one SlideShare presentation and are using the code provided, don't forget to change the `id` for the <`figure`>, <`media`>, and <`flash`> elements each time you add a new video.

5.3 Embedding Prezi Presentations

If you choose to include Prezi presentations in your module, you will need to first copy the information provided in the **Embed** HTML snippet provided by Prezi for the presentation you wish to use. As an example, consider this presentation[13] taken from a presentation developed by UniqU[14] . The Embed snippet for this presentation is as follows:

```
    <div class="prezi-player">
<style type="text/css" media="screen">.prezi-player { width: 550px; } .prezi-player-links { text-align
<object id="prezi_n5c7h_bctk1c" name="prezi_n5c7h_bctk1c" classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553
<param name="movie" value="http://prezi.com/bin/preziloader.swf"/>
<param name="allowfullscreen" value="true"/>
<param name="allowscriptaccess" value="always"/>
<param name="bgcolor" value="#ffffff"/>
<param name="flashvars" value="prezi_id=n5c7h_bctk1c&lock_to_path=1&color=ffffff&autoplay=no"/>
<embed id="preziEmbed_n5c7h_bctk1c" name="preziEmbed_n5c7h_bctk1c"
src="http://prezi.com/bin/preziloader.swf" type="application/x-shockwave-flash" allowfullscreen="true"
allowscriptaccess="always" width="550" height="400" bgcolor="#ffffff"
flashvars="prezi_id=n5c7h_bctk1c&lock_to_path=1&color=ffffff&autoplay=no">
</embed>
</object>
<div class="prezi-player-links">
<p>
<a title="How to think about Connexions!" href="http://prezi.com/n5c7h_bctk1c/">UniqU: Making Connexior
<a href="http://prezi.com">Prezi</a></p></div></div>
```

Since this code is designed for use on websites and not the CNXML language, the majority of this snippet is not necessary. The part that **is** important is the **embed tag** and the **param tags**:

```
    <param name="movie" value="http://prezi.com/bin/preziloader.swf"/>
<param name="allowfullscreen" value="true"/>
<param name="allowscriptaccess" value="always"/>
<param name="bgcolor" value="#ffffff"/>
<param name="flashvars" value="prezi_id=n5c7h_bctk1c&lock_to_path=1&color=ffffff&autoplay=no"/>

<embed id="preziEmbed_n5c7h_bctk1c" name="preziEmbed_n5c7h_bctk1c"
src="http://prezi.com/bin/preziloader.swf" type="application/x-shockwave-flash" allowfullscreen="true"
allowscriptaccess="always" width="550" height="400" bgcolor="#ffffff"
flashvars="prezi_id=n5c7h_bctk1c&lock_to_path=1&color=ffffff&autoplay=no">
</embed>
```

The CNXML example below illustrates how to embed a Prezi player in your module. Notice that it is implemented like a standalone flash object, with a few key features:

- The **src** attribute is set to the **embed** URL for the slideshow, **not the URL for the page on the Prezi website**. The embed URL is the one found in the code snippet as shown above and is of the form "`http://prezi.com/bin/preziloader.swf`).
- Copy all **param** tags from the **embed** snippet.

The final CNXML code for embedding this presentation is as follows:

```
<media id="slidesharemed3452345243523451a" alt="Slide show introducing the ideas behind Connexions.">
```

[13]http://prezi.com/n5c7h_bctk1c/
[14]http://theuniqu.com

```
  <flash id="slidesharefldfadsfasfash"
height="400px" width="550px"
mime-type="application/x-shockwave-flash"
src="http://prezi.com/bin/preziloader.swf">

<param name="movie" value="http://prezi.com/bin/preziloader.swf"/>
<param name="allowfullscreen" value="true"/>
<param name="allowscriptaccess" value="always"/>
<param name="bgcolor" value="#ffffff"/>
<param name="flashvars" value="prezi_id=n5c7h_bctk1c&lock_to_path=1&color=ffffff&autoplay=no"/>

  </flash>
 </media>
```

And results in the following:

This media object is a Flash object. Please view or download it at
<http://prezi.com/bin/preziloader.swf>

Figure 16

TIP: Though the steps described above may look intimidating, the actual process isn't actually that hard. To embed your Prezi presentation, just perform the following steps:

Step 1.Copy the example code (p. 22) above into your module.
Step 2.Replace the URL with the one provided in the <embed> element in the HTML snippet provided by Prezi.
Step 3.Replace the param tags from the example with those that are provided in the <embed> element in the HTML snippet provided by Prezi.
Step 4.Escape the ampersand characters in the URL by replacing `&` with `&amp;`, if any are present.
Step 5.Adjust the height and width parameters to change the size of the presentation (optional).

If you are including more than one Prezi presentation and are using the code provided, don't forget to change the `id` for the `<figure>`, `<media>`, and `<flash>` elements each time you add a new video.