HTML5 Audio How-To

Welcome to HTML5 Audio How-To, where we take you on a journey through using the new HTML5 audio tags now available for use in most modern browsers and mobile platforms.

In the beginning…

Back in the early age of the Internet, a lot of sites were produced with sound effects—although this was seen as trendy, it was something that was often taken to extreme, where unnecessary effects were added that resulted in people being put off from visiting the site! If you wanted to add sound to a web page, you had a number of options, although most meant embedding a plugin to play a sound such as that included with Windows Media Player:

```
<embed type="application/x-mplayer2" pluginspage="http://
www.microsoft.com/Windows/MediaPlayer/" name="mediaplayer1"
ShowStatusBar="true" EnableContextMenu="false" autostart="false"
width="320" height="240" loop="false" src="MyVideo.wmv" />
```

This quickly developed into the battle of the plugins, with the likes of Real Player amongst others, until one emerged as the real victor in 2005: Adobe's™ Shockwave Flash player. It still meant having to put in a complete block of code just to embed a single file though:

```
<object type="application/x-shockwave-flash"
   data="player.swf?audioURL=myAudio.mp3&autoPlay=true"
   height="27" width="320">
   <param name="movie" value="player.swf?audioUrl=myAudio.mp3
      &autoPlay=true">
</object>
```

While this usually worked, it meant needing to use third-party plugins and relying on the visitor's PC (and later laptop) being up to date—not an ideal solution!

The success of Flash was really due to the ubiquity of the plugin (a download option was always available to install the plugin if it wasn't already present), and YouTube, who used it as their delivery mechanism of choice.

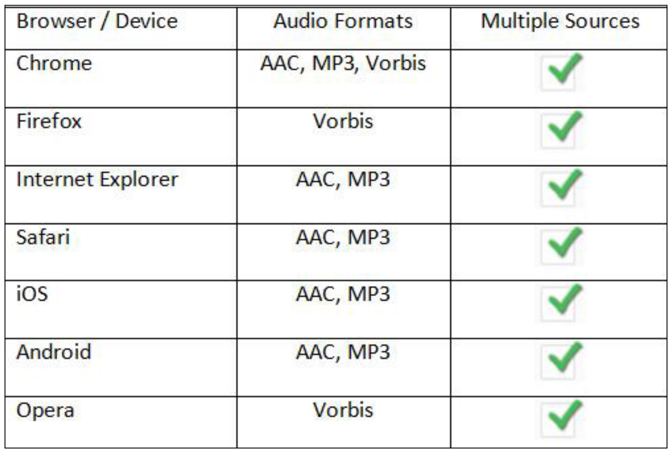
It was not until 2007 that two replacement tags were created—`<audio>` and `<video>`. Their design meant that users didn't have to rely on third-party plugins to get the playback support they needed; with the advent of mobile devices, this was particularly important as it meant that support could be included in the OS and not require any additional configuration. It has even led to Adobe starting to drop support for Flash as the functionality offered in HTML5 is far superior, although support has yet to grow to allow HTML5 to become truly mainstream.

Surely this has to be a good thing, right? Well, yes and no. As you will see in the next recipe, most browsers will play back a HTML5 format audio file, although not all browsers support the same format of audio file!

Browser inconsistencies

The advent of the HTML5 `<audio>` tag has meant that for the first time, audio files can be played in the browser without the need for additional third-party support. Granted, you may not get the same look and feel of a player, depending on which browser you use—the key thing though is that modern browsers could play the files.

This is where the commonality ends though, as software makers haven't been able to agree on a common format to use across all browsers—each supports a different number of formats. In order to illustrate some of the confusion of supporting HTML5 audio, you will, as a developer, need to include a number of formats to maintain support for as wide an audience as possible:



Browser / Device	Audio Formats	Multiple Sources
Chrome	AAC, MP3, Vorbis	✓
Firefox	Vorbis	✓
Internet Explorer	AAC, MP3	✓
Safari	AAC, MP3	✓
iOS	AAC, MP3	✓
Android	AAC, MP3	✓
Opera	Vorbis	✓

The net result is that you will need to cater for two types of audio file formats, namely AAC and Vorbis. MP3 is still available as an audio format, although the quality is not as good, and it has been superseded by the AAC format. It is important to note that although the AAC format is widely referenced, you will also see MP4 or M4A used from time to time; these are valid file extensions for the AAC format.

The files for the tasks in this section are available for download with this PDF, with the exception of the audio files.

Transcoding audio files (must know)

We start this section by getting ready the files we are going to use later on—it is likely you may well have some music tracks already, but not in the right format. We will fix that in this task by using a shareware program called Switch Audio File Converter, which is available from `http://www.nch.com.au/switch` for approximately USD40.

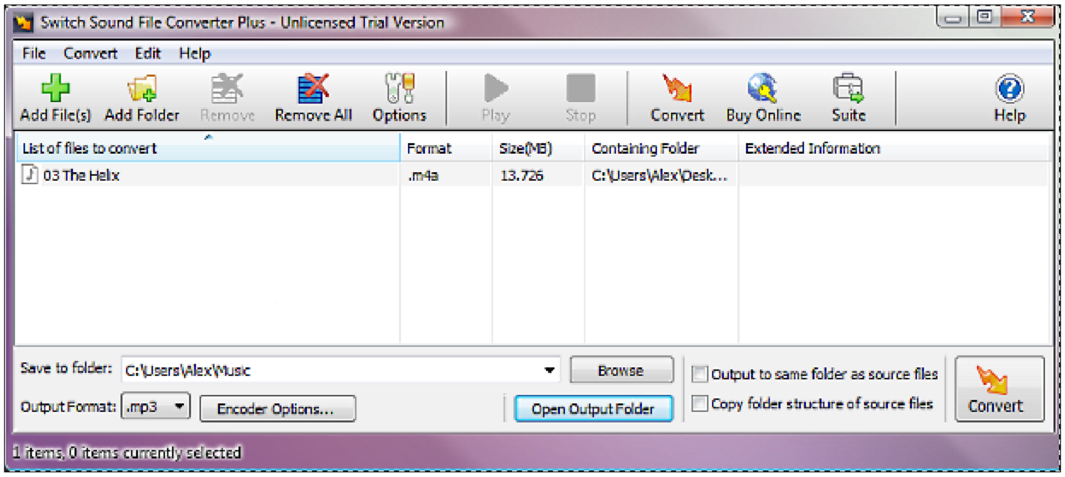
Getting ready

For this task, you will need to download a copy of the Switch Sound Converter application—it is available from `http://www.nch.com.au/switch/index.html`. You may like to note that a license is required for encoding AMR files or using MP3 files in certain instances—these can be purchased at the same time as purchasing the main license.

How to do it...

1. The first thing to do is install the software, so let's go ahead and run `switchsetup.exe`—note that for the purposes of this demo, you should not select any of the additional related programs when requested.

2. Double-click the application to open it, then click on **Add File** and browse to, and then select the file you want to convert:



List of files to convert	Format	Size(MB)	Containing Folder	Extended Information
03 The Helix	.m4a	13.726	C:\Users\Alex\Desk...	

Save to folder: C:\Users\Alex\Music

Output Format: .mp3

1 items, 0 items currently selected

3. Click on **Output Format** and change it to `.ogg`—it will automatically download the required converter as soon as you click on **Convert**. The file is saved by default into your `Music` folder underneath your profile.

How it works...

Switch Sound File Converter has been designed to make the conversion process as simple as possible—this includes downloading any extra components that are required for the purpose of encoding or decoding audio files. You can alter the encoding settings, although you should find that for general use this may not be necessary.

There's more...

There are lots of converters available that you can try—I picked this one as it is quick and easy to use, and doesn't have a large footprint (unlike some others). If you prefer, you can also use online services to accomplish the same task—two examples include Fre:ac (`http://www.freac.org`) or Online-Convert.com (`http://www.online-convert.com`). Note though that some sites will take note of details such as your IP address or what it is you are converting as well as store copies for a period of time.

Installing playback support: codecs (must know)

Now that we have converted our audio files that are ready for playback—it's time to ensure that we can actually play them back in our PCs as well as in our browsers. Most of the latest browsers will play at least one of the formats we've created in the previous task but it is likely that you won't be able to play them outside of the browser. Let's take a look at how we can fix this by updating the codecs installed in your PC.

> For those of you not familiar with codecs, they are designed to help encode assets when the audio file is created and decode them as part of playback. Software and hardware makers will decide the makeup of each codec based on which containers and technologies they should support; a number of factors such as file size, quality, and bandwidth all play a part in their decisions.

Let's take a look at how we can update our PCs to allow for proper playback of HTML5 video.

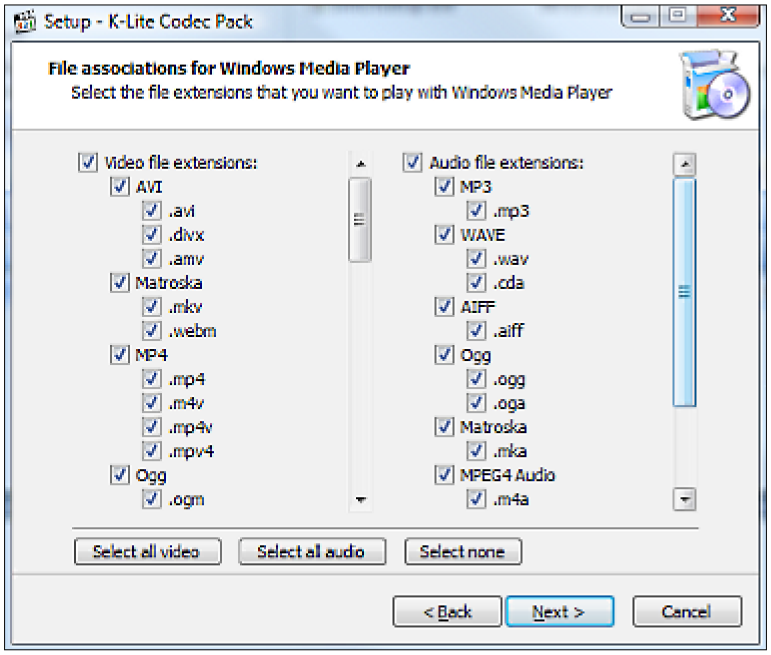
Getting ready

There are lots of individuals or companies who have produced different codecs, with differing results. We will take a look at one package that seems to work very well for Windows, which is the **K-Lite Codec Pack**. You will need to download a copy of the pack, which is available from `http://fileforum.betanews.com/detail/KLite-Codec-Pack-Basic/1094057842/1`—use the blue **Download** link on the right-hand side of the page. This will download the basic version, which is more than sufficient for our needs at this stage.

How to do it...

1. Download, then run **K-Lite_Codec_Pack_860_Basic.exe**. Click on **Next**.
2. On the **Installation Mode** screen, select the **Simple** option.
3. On the **File Associations** page, select **Windows Media Player**.

4. On the File associations screen for Windows Media Player screen, click on **Select all audio**:


Setup - K-Lite Codec Pack

File associations for Windows Media Player

Select the file extensions that you want to play with Windows Media Player

[✓] Video file extensions:	[✓] Audio file extensions:
[✓] AVI	[✓] MP3
[✓] .avi	[✓] .mp3
[✓] .divx	[✓] WAVE
[✓] .amv	[✓] .wav
[✓] Matroska	[✓] .cda
[✓] .mkv	[✓] AIFF
[✓] .webm	[✓] .aiff
[✓] MP4	[✓] Ogg
[✓] .mp4	[✓] .ogg
[✓] .m4v	[✓] .oga
[✓] .mp4v	[✓] Matroska
[✓] .mpv4	[✓] .mka
[✓] Ogg	[✓] MPEG4 Audio
[✓] .ogm	[✓] .m4a

5. On the **Thumbnails** screen, click on **Next**.
6. On the **Speaker configuration** screen, click on **Next**, then **Install**. The software will confirm when the codecs have been installed.

How it works...

In order to play back HTML5 format audio in Windows Media Player, you need to ensure you have the correct support in place; Windows Media Player doesn't understand the encoding format of HTML5 audio by default. We can overcome this by installing additional codecs that tell Windows how to encode or decode a particular file format; K-Lite's package aims to remove the pain of this process.

There's more...

The package we've looked at in this task is only available for Windows so, if you are a Mac user, you will need to use an alternative method. There are lots of options available online—one such option is X Lossless Decoder, available from `http://www.macupdate.com/app/mac/23430/x-lossless-decoder`, which includes support for both `.ogg` and `.mp4` formats.

Installing playback support: mime types

In some instances, you may find that your browser doesn't know how to play back certain audio file types—this usually happens if it can't understand the mime type it needs to use. In this task, we can implement an easy change to rectify this and allow audio files to play correctly.

Getting ready

For this task, we don't need anything except your text editor.

How to do it...

1. Crank up your favorite text editor and create a new text document—save it as `.htaccess` into the root folder of your website.

> You may want to use quotes around the filename when saving it; this forces your text editor to save it as that filename.

2. Add the following text and save the file:

```
AddType audio/aac .aac
AddType audio/mp4 .mp4 .m4a
AddType audio/mpeg .mp1 .mp2 .mp3 .mpg .mpeg
AddType audio/ogg .oga .ogg
AddType audio/wav .wav
AddType audio/webm .webm
```

3. Try playing the audio file now—hopefully you may find nothing has changed and that the video still continues to play. The change you've made now ensures that anyone playing the video will be able to play it properly depending on which browser they use.

How it works...

In most instances, browsers will usually play the right video. However, the HTTP protocol doesn't know about the concept of file extensions—this means you cannot rely on a browser being able to play the right audio file all of the time. To get around this, we add this small configuration change so that the browser is able to correctly determine the file format from the extension and play the audio track.

There's more...

You will notice though from the list above that there seems to be a fair number of audio file formats in relation to audio's sister tag, the `<video>` tag. The audio formats are still very fluid, as browsers have yet to standardize on any one format. You should note though that two formats:

```
AddType audio/ogg .oga
AddType audio/wav .wav
```

...are unnecessary and don't need to be used:

- The `.wav` format produces large files that are slow to play back
- `Xiph.org` (who maintain the OGG format), have stipulated that the `.ogg` format should be used instead (it brings the file extension in line with its video equivalent, `.ogg`).

These should only be used if absolutely necessary—this should be an exception rather than the rule!

Embedding HTML5 audio (must know)

Now that we are set up to play audio files correctly, let's take a look at how you embed those same files into your web pages. Remember how we looked at the embedding of audio files before the advent of the HTML5 `<audio>` tag? In this exercise, we're going to focus on how you would embed the same files using the new tags.

Getting ready

This is an easy exercise—all you need is your text editor and your audio file in a number of different formats. For the purposes of this exercise, we will use two formats—OGG and MP4.

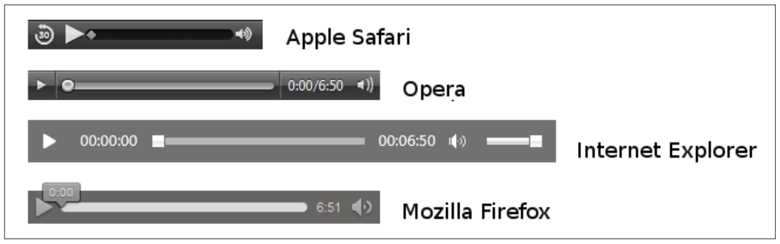
How to do it...

1. Open up your usual text editor and copy in the following code:

```
<!doctype html>
<html>
<head>
    <title>Audio Element Sample</title>
</head>
<body>
    <h1>Audio Element Sample</h1>
    <audio controls>
```

```
            <source src="03 The Helix.m4a" type="audio/mp4">
            <source src="03 The Helix.ogg" type="audio/ogg">
            HTML5 audio not supported
        </audio>
    </body>
    </html>
```

2. Save this as `audiodemo.html`—you can now play this in your browser, where you will see a player appearing similar to the following screenshot:



How it works...

This code is only effective in modern browsers—each browser replaces the `<audio>` tags with its own implementation of an audio player. By default, each browser automatically ignores code that it doesn't understand; this means that if, for example, it can't understand the MP4 format, then it will automatically switch to using the OGG format instead. We've also added the `controls` attribute in the code—this means that you will be able to control actions such as volume, playback, or the pausing of audio as well as fast-forwarding or rewinding the track being played.

There's more...

The previous code is the basic code required to embed audio tracks into HTML5—if desired, you can further customize the code by adding any of the following attributes such as:

Attribute:	**Description:**
src	Provides the URL of the audio file
autoplay	Indicates that the audio file begin automatically, where possible. The value can be set in one of the following ways: <audio autoplay> <audio autoplay="autoplay"> <audio autoplay=""> Note: using autoplay is not recommended unless necessary, as it will use bandwidth that may be limited.

Attribute:	**Description:**
controls	Tells the browser to display its default audio control set, where possible. It can be set in one of three ways: <audio controls> <audio controls="controls"> <audio controls="">
loop	Indicates that the audio should be played continuously in a loop. It can be set in one of three ways: <audio loop> <audio loop="loop"> <audio loop="">
preload	Suggests to the browser how it should attempt to preload the audio file. It can have one of three possible values: ▸ none: don't perform any preloading ▸ **metadata**: only load the audios ▸ **auto**: lets the browser decide for itself (this is set by default)

Support for cross-browser playback (must know)

In the previous task, we focused on how to add audio to your pages using the new `<audio>` tag. This leaves one crucial drawback though—many people are still using older browsers, which don't offer native support for the tags. In order to get around this, we have to make a change to the code from the previous exercise. This is the subject of our next task.

Getting ready

For the purposes of this task, you need to avail yourself of a suitable player in SWF format; one will be included with the code download available for this recipe. We will also need the MP4 and OGG format files that you created in previous exercises—you will need to alter the file names shown below accordingly, to match your own.

How to do it...

1. Crack open your usual text editor and add the following code from the previous exercise:

```
<!doctype html>
<html>
<head>
    <title>Audio Element Sample</title>
</head>
<body>
    <h1>Audio Element Sample</h1>
    <audio controls>
        <source src="Helix.m4a" type="audio/mp4">
        <source src="Helix.ogg" type="audio/ogg">
        HTML5 audio not supported
    </audio>
</body>
</html>
```

2. We need to add the fallback support, so add the following code just below the second `<source>` tag:

```
<audio controls>
   <source src="Helix.ogg" type="audio/ogg">
   <source src="Helix.m4a" type="audio/mp4">
   <object type="application/x-shockwave-flash"
      data="player.swf?audioUrl=Helix.mp4&autoPlay=true">
   <param name="movie"
      value="player.swf?audioUrl=Helix.mp4&autoPlay=true">
   </object>
   <a href="Helix.mp4">Download the audio file</a>
</audio>
```

How it works...

The principle is simple—it works on the basis that although the code may show in browsers, it will be ignored if the browser doesn't understand it. Here, we start with the OGG format file (supported by Firefox, Opera, and Chrome); if the browser doesn't understand this format, then it will attempt to play the MP4 format (this will likely be Chrome, Safari, or IE). If all else fails, then the browser will play the MP4 file as an embedded SWF or show a link to allow the user to download and play the file locally.

There's more...

While HTML5 audio support is still in a state of flux, it is crucial to ensure that you include some form of fallback support for those browsers that don't support the `<audio>` tags. Browsers will automatically ignore those tags they don't understand—the best fallback support to use is Flash's SWF format, as WAV is supported by browsers, but not recommended. It doesn't compress very well as a format, which results in large files.

Using an existing audio library (must know)

Now that we have looked at the basics of embedding audio files using the new tags, it's time to focus on how you can use one of the prebuilt libraries available that provide a customized version. We're going to take a look at one called MediaElementJS, available from `http://www.mediaelementjs.com`; this is an open source library that covers both the new `<video>` and `<audio>` tags.

Getting ready

For this exercise, you will need to download a copy of the library and then extract the following files: `controls.png`, `jquery.js`, `mediaelement`-and-`player.min.js`, and `mediaelementplayer.css`. You will also need copies of your audio files, these should be in MP4 and OGG formats.

How to do it...

1. Open up your favorite text editor, then copy in the code from the *Embedding HTML5 Audio* task from earlier in this recipe, you should have:

```
<!doctype html>
<html>
<head>
    <title>Audio Element Sample</title>
</head>
<body>
    <h1>Audio Element Sample</h1>
    <audio controls>
        <source src="03 The Helix.m4a" type="audio/mp4">
        <source src="03 The Helix.ogg" type="audio/ogg">
        HTML5 audio not supported
    </audio>
</body>
</html>
```

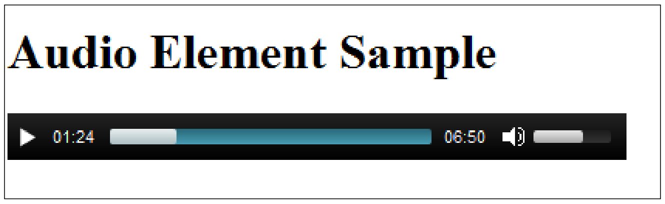
2. We need to add links to the MediaElementJS library, so go ahead and modify the code as follows:

```
<title>Audio Element Sample</title>
<code>
   <script src="build/jquery.js"></script>
   <script src="build/mediaelement-and-player.min.js">
      </script>
   <link href="build/mediaelementplayer.css"
      rel="stylesheet" />
</code>
```

3. Add the following code next, below the closing `</audio>` tag:

```
 <script>
   // using jQuery
   $('audio').mediaelementplayer();
</script>
```

4. If all is well, you should have something looking like the following screenshot:



How it works...

The key to this is that we've kept the base code untouched in terms of the `<audio>` tags—MediaElementJS is different in that it doesn't use a fallback if the browser doesn't understand the tags but a *fallforward*. It works by using a Flash file as the lowest common denominator, then progressively pushing to using any of the supplied audio files depending on the capabilities of the browser. There are other libraries available, although not all operate in the same way; you may want to take a look at something like Speakker (`http://www.speakker.com`), or for a more bare-bones system, try `Audio.js` (`http://kolber.github.com/audiojs/`).

There's more...

The observant amongst you will spot a problem with our code—where's the real fallback option? It's a good question! In order for the previous code to be really successful, you will need to modify it as follows, using the code from the previous exercise:

```
<audio>
   <source src="03 The Helix.m4a" type="audio/mp4">
   <source src="03 The Helix.ogg" type="audio/ogg">
   <object type="application/x-shockwave-flash"
      data="player.swf?audioUrl=myAudio.mp3&autoPlay=true">
   <param name="movie"
      value="player.swf?audioUrl=myAudio.mp3&autoPlay=true">
   </object>
   <a href="myAudio.mp3">Download the audio file</a>
</audio>
```

Building a basic audio player in jQuery (become an expert)

In the previous task, we took a look at using a prebuilt library to provide the audio functionality—while this works well, there is one downside: you are limited to the styling that comes by default with the library, which may not necessarily fit in with your site's style. We're going to fix that. In this task and the next, we will look at the code required to produce your own custom player using the MediaElementJS backend functionality.

Getting ready

For this task, you will need the same files as you had from the previous exercise—this includes the jQuery library, the MediaElementJS files, and your audio files.

How to do it...

1. Let's begin by setting up the basic framework. Go ahead and copy the following code into a new document and save it as `audiodemo.html`:

   ```
   <!doctype html>
   <html>
   <head>
       <title>Audio Element Sample</title>

      <code>
      </code>
   ```

```
</head>
<body>
</body>
</html>
```

2. We need to start adding our JavaScript library calls, so go ahead and add this into your `<head>` tags:

```
      <script src="https://ajax.googleapis.com/ajax/libs/
jquery/1.7.2/jquery.min.js"></script>
      <script src="js/mediaelement-and-player.js"></script>
     <link href="css/stylenew.css" rel="stylesheet" type="text/
css" />
```

3. Next comes the HTML framework for the audio player. Add the following, immediately after the opening `<body>` tag:

```
    <div class="audio-player">
      <h1>Robert Vadney - The Helix</h1>
      <img class="cover" src="img/vespera.jpg" alt="">
      <audio id="audio-player" controls="controls">
         <source src="media/Helix.m4a" type="audio/mp4">
         <source src="media/Helix.ogg" type="audio/ogg">
         HTML5 audio not supported
      </audio>
   </div>
```

4. Last, but by no means least, here's the JavaScript functionality to initialize the player; add the following code immediately below the last block of code, and before the closing `</body>` tag:

```
    <script>
      // using jQuery
        $(document).ready(function() {
            $('#audio-player').mediaelementplayer({
                alwaysShowControls: true,
                features: ['playpause','volume','progress',
                'current', 'duration'],
                audioVolume: 'horizontal',
                audioWidth: 50,
            audioHeight: 50
         });
      });
   </script>
```

You won't see a great deal working—while the main functionality is in place, and ready to be used, not much will be shown until you add the CSS styling.

How it works...

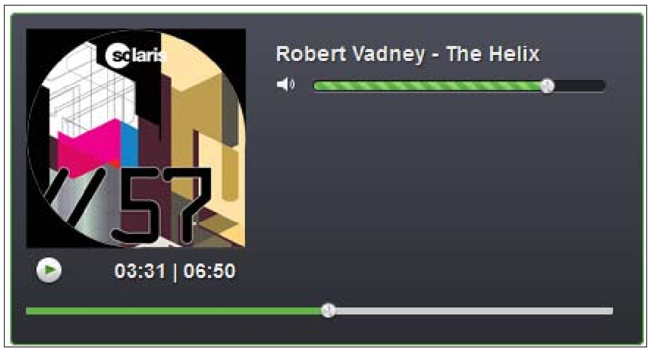
We've used the same basic HTML code throughout this recipe to implement an audio player—MediaElementJS replaces the script calls with its code for the player that you see on screen. In this instance, we've gone a step further in comparison to the previous exercise by specifying a number of additional parameters within MediaElementJS. This will give us a little more control over the features that are displayed—which will begin to take shape once we add the requisite CSS styles, which will be the subject of our next task.

Styling our player (must know)

In this task, we take care of the code required to display and use our player—this is where you will see the features take shape and begin to be able to use the player.

Getting ready

There is nothing special required for this player, all you need is your text editor. We are going to produce something like the following screenshot as a player—this will work in most modern browsers:



How to do it...

1. There's a lot of CSS code to go through this exercise, so let's begin by creating a stylesheet, which we will call `style.css`.
2. Add the following to take care of the main image and basic frame:

```
.cover { padding: 10px; }

.audio-player {
    width: 430px; border: 1px solid #64B44C; margin: auto;
      margin-top: 30px; background: #4c4e5a;
    background: -webkit-linear-gradient(top, #4c4e5a 0%,
      #2c2d33 100%);
    background: -moz-linear-gradient(top, #4c4e5a 0%, #2c2d33
      100%);
    background: -o-linear-gradient(top, #4c4e5a 0%, #2c2d33
      100%);
    background: -ms-linear-gradient(top, #4c4e5a 0%, #2c2d33
      100%);
    background: linear-gradient(top, #4c4e5a 0%, #2c2d33,
      100%);
    -webkit-border-radius: 4px; -moz-border-radius: 4px;
    border-radius: 4px;
}

.audio-player h1 { margin: 0; padding: 0; border: none;
    outline: none; font-family: Helvetica, Arial, sans-serif;
    font-weight: bold; font-size: 14px; color: #ececec;
    text-shadow: 1px 1px 1px rgba(0,0,0, .5);
    width: 250px; position: absolute; float: left; top: 50px;
 margin-left: 180px;
}
```

3. It's now time to add the styles for the controls, take note: there is a lot of code required here:

```
.audio-player button { margin: 0; padding: 0; border: none;
    outline: none; }

.mejs-controls .mejs-play button, .mejs-controls .mejs-pause
button {
    background: url("../img/play-pause.png") repeat scroll 0 0
      transparent;
    float: left;
    height: 21px;
    margin-left: 15px;
    margin-top: -10px;
    width: 21px;
}

.mejs-controls .mejs-pause button { background-position:0
    -21px; }

.mejs-controls .mejs-mute button, .mejs-controls .mejs-unmute
button {
    background: url("../img/mute-unmute.png") repeat scroll 0
    0 transparent; float: left; height: 12px; margin-left:
    145px; margin-top: -132px; position: absolute; width:
    14px; }

.mejs-controls .mejs-time-rail .mejs-time-handle, .mejs-controls
.mejs-horizontal-volume-slider .mejs-horizontal-volume-handle {
    background: url("../img/handle.png") no-repeat scroll 0 0
    transparent; display: block; height: 14px; position:
    absolute; width: 12px; }

.mejs-time-handle { top: -4px; }

.mejs-horizontal-volume-handle { top: -2px; }

.mejs-controls div.mejs-time-rail { width: 400px; }

 .mejs-controls .mejs-time-rail span { left: 0; bottom: 0;
    position: absolute; display: block; width: 400px;
    height: 5px; cursor: pointer;
    -webkit-border-radius: 0px 0px 2px 2px;
    -moz-border-radius: 0px 0px 2px 2px;
    border-radius: 0px 0px 2px 2px; }

.mejs-controls .mejs-time-rail .mejs-time-total { background:
    #999999; }

.mejs-controls .mejs-time-rail .mejs-time-loaded { width: 0;
    background: #cccccc; }

.mejs-controls .mejs-time-rail .mejs-time-current { width: 0;
    background: #64b44c; }

 .mejs-controls .mejs-time-rail .mejs-time-float {
    position: absolute; display: none; width: 33px; height:
    23px; top: -26px; margin-left: -17px; background:
    url(../img/time-box.png); }

.mejs-controls .mejs-time-rail .mejs-time-float-current {
    width: 33px; display: block; left: 0; top: 4px;
    font-family: Helvetica, Arial, sans-serif; font-size:
    10px; font-weight: bold; color: #666666; text-align:
    center; }
```

```
.mejs-controls .mejs-horizontal-volume-slider {
    cursor: pointer; float: left; margin-left: 205px;
    margin-top: -130px; position: absolute; }

.mejs-controls .mejs-horizontal-volume-slider .mejs-horizontal-
volume-total { width: 200px; height: 8px;
    background: #212227;
    -webkit-box-shadow: inset 0px 1px 0px rgba(0,0,0, .3), 0px
      1px 0px rgba(255,255,255, .25);
    -moz-box-shadow: inset 0px 1px 0px rgba(0,0,0, .3), 0px
      1px 0px rgba(255,255,255, .25);
    box-shadow: inset 0px 1px 0px rgba(0,0,0, .3), 0px 1px 0px
      rgba(255,255,255, .25);
    -webkit-border-radius: 6px; -moz-border-radius: 6px;
    border-radius: 6px; }

.mejs-controls .mejs-horizontal-volume-slider .mejs-horizontal-
volume-current { background: url("../img/volume-
    bar.png") repeat-x scroll 0 0 transparent;
    border-radius: 6px 6px 6px 6px; height: 6px; left: 1px;
    position: absolute; top: 1px; width: 0; }

.mejs-time { float: left; margin-left: 70px; margin-top: -8px;
    position: absolute; color: #ECECEC; font-family:
    Helvetica,Arial,sans-serif; font-size: 14px;
    font-weight: bold; position: absolute;
    text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.5); }

.mejs-controls div.mejs-time-rail { float: left; margin-left:
10px; margin-top: 30px; position: absolute; width: 400px; }
```

How it works...

Although it seems like a lot of code, all we have done here is to take each individual element of the player and add a style to it. We've used a fair number of CSS3 styles in this code; as the HTML5 `<audio>` tags are only really designed to work in the most modern browsers, it seems sensible to take advantage of the newer styles!

Using HTML5 audio players in a CMS system (must know)

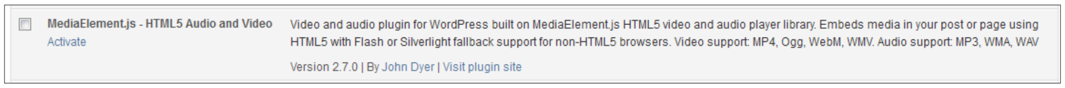
Throughout this recipe, our focus has been on adding audio files to static web pages, but with increasing numbers of people using content management systems, it's important to take a look at how you could add audio to these systems. In this task, we're going to take a look at how to achieve this. We will use WordPress as an example, although the same principles will apply to other systems as well.

Getting ready

For this task, you will need access to a working installation of WordPress—I use a standalone version (currently Version 3.3.2, at time of writing), available from `http://www.wordpress.org`. You will also need to avail yourself of a copy of the MediaElementJS—the plugin for use in WordPress that is available from `http://wordpress.org/extend/plugins/media-element-html5-video-and-audio-player/`.

How to do it...

1. Download a copy of the WordPress plugin, then upload the `media-element-html5-video-and-audio-player` folder to the `/wp-content/plugins/` directory of your WordPress installation.
2. You need to log in to the administration section of WordPress, then click on the **Plugins** tab.
3. You will see the plugin listed. Go ahead and click on **Activate** to make the plugin live:



MediaElement.js - HTML5 Audio and Video
Activate

Video and audio plugin for WordPress built on MediaElement.js HTML5 video and audio player library. Embeds media in your post or page using HTML5 with Flash or Silverlight fallback support for non-HTML5 browsers. Video support: MP4, Ogg, WebM, WMV. Audio support: MP3, WMA, WAV

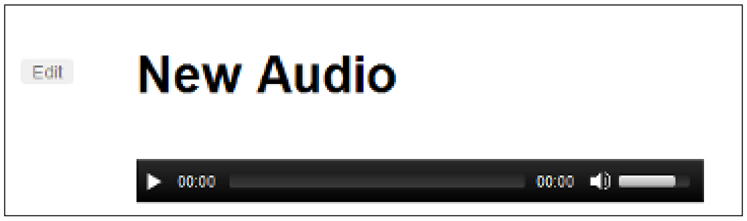
Version 2.7.0 | By John Dyer | Visit plugin site

4. Now that the plugin is live, go ahead and create a new page in your WordPress installation. Enter the following within the editor:

```
[audio mp4="http://127.0.0./wordpress/audio/helix.ogg"

   ogg="http://127.0.0./wordpress/audio/helix.ogg"]
```

5. On the page you've created, you will see something similar to the following screenshot:



How it works...

The plugin uses WordPress' shortcode technology—WordPress replaces any instances of a specific keyword with a predetermined phrase or block of code. Here, the plugin's author has created two shortcodes, but with a number of different options attached to them—one each for audio and video. For example, the following is a shortcode example for audio:

```
[audio mp3="http://mysite.com/mymedia.mp3"
   ogg="http://mysite.com/mymedia.ogg" preload="true"
   autoplay="true"]
```

The following shortcode example uses the same principles, but this time for video:

```
[video mp4="http://mysite.com/mymedia.mp4"
   ogg="http://mysite.com/mymedia.ogg"
   webm="http://mysite.com/mymedia.webm"
   poster="http://mysite.com/mymedia.png" preload="true"
   autoplay="true" width="640" height="264"]
```

To give you some idea of why it is far easier to use shortcodes, the following is the block of code that is produced when using the shortcode example from our task. As you can see, there is a fair amount of code:

```
<div id="mep_0" class="mejs-container mejs-player mejs-  audio"
style="width: 400px; height: 30px;">
   <div class="mejs-inner">
      <div class="mejs-mediaelement">
         <audio id="wp_mep_1" class="mejs-player " data-
mejsoptions="{"features":["playpause","current","progress",
"duration","volume","tracks","fullscreen"],"audioWidth":400,
"audioHeight":30}" preload="none" tabindex="0"
src="http://127.0.0./wordpress/audio/helix.ogg">
            <source type="audio/mp4" src="http://127.0.0./wordpress/
audio/helix.ogg"></source>
```

```
            <source type="audio/ogg" src="http://127.0.0./wordpress/
audio/helix.ogg"></source>
            <object width="400" height="30" data="http://127.0.0.1/
wordpress/wp-content/plugins/media-element-html5-video-and-audio-
player/mediaelement/flashmediaelement.swf" type="application/x-
shockwave-flash">
               <param value="http://127.0.0.1/wordpress/wp-content/
plugins/media-element-html5-video-and-audio-player/mediaelement/
flashmediaelement.swf" name="movie">
               <param value="controls=true&file=http://127.0.0./
wordpress/audio/helix.ogg" name="flashvars">
            </object>
         </audio>
      </div>
   <div class="mejs-layers">
      <div class="mejs-poster mejs-layer" style="display: none; width:
400px; height: 30px;"></div>
   </div>
   <div class="mejs-controls">
      <div class="mejs-button mejs-playpause-button mejs-play">
         <button title="Play/Pause" aria-controls="mep_0"
type="button"></button>
      </div>
      <div class="mejs-time mejs-currenttime-container">
         <span class="mejs-currenttime">00:00</span>
      </div>
      <div class="mejs-time-rail" style="width: 218px;">
         <span class="mejs-time-total" style="width: 208px;">
      </div>
      <div class="mejs-time mejs-duration-container">
         <span class="mejs-duration">00:00</span>
      </div>
      <div class="mejs-button mejs-volume-button mejs-mute">
         <button title="Mute Toggle" aria-controls="mep_0"
type="button"></button>
      </div>
      <div class="mejs-horizontal-volume-slider mejs-mute">
         <div class="mejs-horizontal-volume-total"></div>
         <div class="mejs-horizontal-volume-current" style="width:
40px;"></div>
         <div class="mejs-horizontal-volume-handle" style="left:
27px;"></div>
      /div>
   </div>
      <div class="mejs-clear"></div>
   </div>
</div>
```

If you are using WordPress, and want to use this plugin, then it is worth visiting `http://wordpress.org/extend/plugins/media-element-html5-video-and-audio-player/`, for a full list of options available for this plugin. Other plugin options for MediaElementJS will have similar documentation available.

Phew! That was a real whistle-stop tour of the `<audio>` plugin, wasn't it? We've had a look at a number of things throughout this recipe, including how to:

- Convert audio files into the correct format for HTML5
- Understand some of the browser inconsistencies of the `<audio>` format
- Manually embed audio files into your pages as well as use existing audio libraries
- Provide fallback support
- Embed the `<audio>` tags within a content management system such as WordPress

I hope you've enjoyed reading this recipe, and completing the demos, as much as I have had writing them—they have only just scratched the surface of what is possible with the `<audio>` tags. There is a lot more you can do, including using other functionality such as WebGL or SVG graphics. Here's to the adventure!