15 Incorporating Sound and Video

In this chapter, you will learn how to

- ✔ Understand the purpose and scope of the new <*audio*> and <*video*> tags in HTML5.
- ✔ Play multimedia types and choose formats and codecs.
- ✔ Use the <*video*> tag.
- ✔ Use the <*audio*> tag.

Playing video and audio on the Web is a bit more difficult than other Web-related tasks. This stems from the multitude of formats that are available from competing vendors and open source groups. These formats have varying degrees of support in the popular modern Web browsers; often they have no support at all in older browsers. Together, these factors make it difficult to deliver audio and video that's consistently playable for all of your visitors.

The addition of the <*video*> and <*audio*> tags in HTML5 makes the process of delivering and playing video and audio more straightforward. Playing multimedia will get easier over time as newer browsers support the tags and people upgrade their older browsers. However, for the foreseeable future, it will still be necessary to encode your multimedia files into multiple formats.

See Also Do you need only a quick refresher on the topics in this chapter? See the Key Points section at the end of this chapter.

Practice Files Before you can use the practice files provided for this chapter, you need to install them from the book's companion content page to their default locations. See "Using the Practice Files" in the beginning of this book for more information.

What's New with Audio and Video in HTML5?

Traditionally, developers and designers have most commonly set up pages to play video and audio on the Web using Adobe Flash. Sites such as YouTube (*http://www.youtube.com*) embed video inside of a Flash file. This requires that the end user has the Adobe Flash player installed.

The HTML5 specification introduces an alternative to that: a standard tag, <*video*>, which enables the playing of video content. However, the <*video*> tag still requires a video file and also requires end users to have an appropriate player installed on their computers.

For audio clips, the traditional delivery method has been to use the <*object*> or <*embed*> tag to embed a clip on a page. HTML5 provides the <*audio*> tag to do this instead.

As of this writing, the <*audio*> and <*video*> tags have limited support in Web browsers. Adding to the complexity is the need to support multiple formats for video, depending on what your visitor's browser can play.

This chapter shows you how to take advantage of the new <*audio*> and <*video*> tags and helps to sort out the difficulties surrounding video compatibility. Before going further, you should understand that at the time of this writing, support for these two new tags is limited to the following browsers:

- Internet Explorer 9+
- Firefox 3.5+
- Safari 3+
- Chrome 3+
- Opera 10.5+
- iPhone 1+
- Android 2+

Browsers that don't support these tags ignore them, but if you want to deliver your audio or video to a full range of browsers—new and old, you'll need to be able to do it without the use of these tags. This chapter shows both the old and new methods.

HTML Multimedia Basics

Before getting into the details of creating multimedia-rich Web pages, you should have a basic understanding of how HTML5—and previous versions of HTML—present audio and video clips.

The most common method of placing multimedia content on a Web page is to *embed* an audio or video clip in the page so that it plays within the page itself when the visitor clicks a button. For this to work, visitors to your site must be using a Web browser that supports the type of sound or video file you're providing, or they must download and install a plug-in (a helper program) to add support for that file to their browser. If your audience uses Microsoft Internet Explorer version 5.5 and higher, you can use the <*object*> tag for this; otherwise you use the <*embed*> tag. Or, if your audience uses an HTML5-compliant browser, you can use the new <*audio*> and <*video*> tags for this.

As an alternative, you can *link* to an audio or video clip so that it plays in an external application (such as Microsoft Windows Media Player) when the visitor clicks its hyperlink. For this to work, the visitor must have an external application that supports the type of sound or video file you're providing, or they must download and install a separate program. This technique works the same in all browsers, though, which is a plus. Use the <*a*> tag for the link, just like with any other hyperlink. For example:

```
<a href="mysong.mp3">Playing my song!</a>
```

This chapter focuses mainly on the embedding type of multimedia presentation.

Multimedia Formats and Containers

Discussion of multimedia on the Web must begin with an understanding of the different formats. When people talk about video files, they're usually talking about files with an .avi, .mp4, or .mkv extension. These extensions are simply indicators of the container format for the video file itself; they don't indicate the format in which the video was encoded.

There are several common container formats, including Ogg (.ogv), Flash Video (.flv or .f4v), the aforementioned Audio Video Interleave (.avi), MPEG-4 Part 14 (.mp4), Matroska (.mkv), and many others. See *http://en.wikipedia.org/wiki/Container_format_%28digital%29* for an overview of container formats.

Additionally, video files almost always contain audio tracks. The container file includes both the video and audio components.

There is also a new format, called WebM, which is similar to Matroska. WebM is an open-source video container format that will likely grow in popularity due in part to its support by Google. WebM is meant to be used exclusively with the VP8 video codec and the Vorbis audio codec (more on codecs in the next section).

Codecs: Decoding the Video and Audio

When a producer (the person or organization making the audio or video available) encodes multimedia, they choose the format in which to encode the file. The person who views that video or listens to the audio must have the appropriate decoding software installed on their computer to play the file. This decoding software is called a ***codec***.

You'll see the word codec in this chapter and in other publications about video and audio. The word itself is shorthand for encode/decode (or decode/encode depending on whom you ask). The codec refers to the style in which the video or audio file was encoded or formatted. To decode a video or audio file means that the computer uses an algorithm to decipher the encoded video or audio into a human-consumable form.

Now throw in the Web browser. The browser, such as Internet Explorer, either needs to have built-in support for a format or needs to have a plug-in available to recognize that it can play the audio or video file. Luckily, all of the common formats and codecs today are either supported natively or are readily available in some form of plug-in installer for the popular Web browsers. As newer browsers that support HTML5 appear, the use of specific third-party plug-ins—at least for video and audio—will (hopefully) become a thing of the past.

Just as there are numerous container formats, there are also several common video encoding formats. Some of the most popular ones include H.264, VP8, DivX, Theora, and several others. If you plan to do much video work on the Web, you'll likely need to account for several different formats and containers to reach the widest possible audience.

As with video, playing audio through a computer or hand-held mobile devices (such as SmartPhones) requires a codec to read the file and play it back. Two popular formats are MPEG-4 Audio Layer 3, which you might recognize as MP3, and AAC, frequently used by Apple. Other formats include Vorbis, which is frequently used in an Ogg container.

Many of the video formats support ***profiles***, which are essentially the parameters used when the video is encoded. For example, a high profile H.264 video provides higher quality but at the cost of a much larger file size—too large for general use on the Web. For now, it's sufficient to know that different profiles exist, and different profiles are appropriate for different applications.

Which Format to Choose?

If all of this sounds complex, it is. Not only is it tough to choose among the multiple formats, but whatever your choice, there's no guarantee that your visitors will be able to play that format anyway. At a high level, audio is easier than video, so for all intents and purposes, your energy will be put into working with video formats.

So how do you choose which format to use? The answer is that you don't choose one format; you choose three or four. The ultimate goal is to make the video available to the widest possible audience. With that in mind, you will need to be able to convert a source video file to several formats to ensure that visitors can play it.

Table 15-1 shows the three primary containers that you'll use, not including Flash.

TABLE 15-1 Common Video Formats for the Web

Container	Video Codec	Audio Codec
Ogg	Theora	Vorbis
mp4	H.264	AAC
WebM	VP8	Vorbis

As of this writing, Microsoft Internet Explorer 9 supports the <*video*> tag, but it only supports the H.264 video format. Previous versions of Internet Explorer don't support the <*video*> tag, but don't worry; you'll see how to work around that restriction a bit later in this chapter.

Mozilla Firefox versions 3.5 and later support the WebM and Ogg containers. Safari supports H.264 video and AAC audio in an mp4 container. Opera supports WebM and Ogg containers as well. The Ogg container will almost certainly contain Theora video and Vorbis audio.

File Size and Quality

The word "size" has two meanings for a video clip: the file size and the display size (the number of pixels vertically and horizontally). As you might expect, these two factors are related—the larger the clip's display window, the larger the file size. A clip on a Web page need not fill the entire monitor; a window of two to three inches is usually sufficient.

The display size is not the only determinant of the file size, however. Some file formats are smaller than others because they use varying degrees of compression to decrease their file sizes. A video clip is compressed using a certain compression *algorithm*, which is a set of math formulas used to remove excess space in the clip for storage. To play a compressed clip, the computer playing it must possess an appropriate codec.

Note A compression algorithm works by identifying repeated characters or patterns in the data file and substituting more compact codes for them. For example, an algorithm might change 00000000000000000000 to something like 20*0.

Further, video clips vary according to the number of frames per second (fps); more frames per second means smoother playback and larger file size. A VHS videotape records at 30 fps, but for Web use, a frame rate of 15 fps works well because it results in a much smaller file size. You can set the number of frames per second when you record the video clip, or use a third-party program to decrease the frames per second of a pre-recorded clip.

When a sound clip is digitized (converted to digital format), a series of sound "snapshots" are taken per second. These snapshots are called *samples*. Higher *sampling rates* (the number of samples per second) yield higher sound reproduction accuracy, but at the cost of larger file sizes. Sampling rates for audio clips are measured in kilohertz—for example, 11 KHz, 22K Hz, or 44 KHz.

Note "Kilo" means thousand; an 11 KHz clip contains approximately 11,000 samples per second.

Sound clips also have varying *sample resolutions*, which are the number of bits used to describe each sample. Common sample resolutions are 8-bit, 16-bit, and 32-bit. The more bits that are sampled, the larger the file will be.

Sound clips can be recorded in either mono or stereo, referring to the number of audio channels in the recording. Mono uses a single channel, which is duplicated in each speaker. Stereo uses two channels, with one channel playing back in each of two speakers. Stereo clips are approximately double the file size of mono ones.

When recording audio clips, you can usually choose between various sampling rates and resolutions. Here are some of the most common combinations of settings:

Settings	Quality
8 KHz, 8-bit, mono	Telephone quality
22 MHz, 16-bit, stereo	Radio quality
44 KHz, 16-bit, stereo	CD quality

Encoding Video

Now that you have a high-level view of video and audio playback on the Web, you might be wondering how you encode your favorite vacation videos into three formats (four if you count Flash). The clips provided for the exercises in this chapter are ready to go, but you will need to prepare your own video clips on your own.

Just as playback is complex, so too is encoding. People frequently employ a combination of software to encode and convert videos between formats. For example, software called Handbrake is popular for converting video to H.264 and AAC format for playback on Apple devices, and is also useful for converting video for the Web.

Converting to an Ogg Theora video with Vorbis audio can be accomplished using several different software packages including ffmpeg2theora, VLC media player, Firefogg (a plug-in for Firefox), and others. Firefogg, ffmpeg, and several others can also convert to WebM format.

Tip If you're using Firefox (or want to encode video), a simple and effective way to do so is to use VLC. Be prepared to wait, though. Converting videos between formats can be a slow process. I used VLC for all the conversions made while writing this chapter.

With the goal of making video on your site widely available, you'll typically encode your videos into each of these three formats as well as Flash. Using those four formats makes the video natively available in new browsers with built-in support for the new <*video*> and <*audio*> tags but still makes Flash available for visitors with older browsers.

Embedding Video Clips

So far, you've seen a lot of background material for something that seems like it should be easy! And to think we've only scratched the surface. This section shows how to use the <*video*> tag to place video on a page as well as how to fall back to Flash video if necessary.

Introducing the <video> Tag

At a basic level, the <*video*> tag looks like this:

```
<video src="myvideo.ogv"></video>
```

There are several attributes and different ways to use the <*video*> tag that make it more configurable for your needs and the needs of your audience. Several attributes are helpful, including:

- *autoplay*
- *controls*
- *height*
- *loop*
- *preload*
- *width*

Not surprisingly, you use the *width* and *height* attributes to set the width and the height of the video display area on the page, as shown in the following example:

```
<video src="myvideo.ogv" width="320" height="240"></video>
```

The *controls* attribute determines whether a default set of playback controls should be visible within the browser. This can be helpful and I recommend using it. In fact, if you don't use the controls attribute, the visitor has no way to replay the video without reloading the entire page. How annoying! Here's an example of the *controls* attribute:

```
<video src="myvideo.ogv" controls></video>
```

The *preload* attribute tells the browser to begin downloading the video immediately when the element is encountered. If the video is the central theme of the page, and it's likely that all (or most) visitors will want to watch the video, then it's a good idea to use the preload option. However, if the video element is a small part of the page and visitors aren't likely to watch it, then preloading the video is just a waste of bandwidth. Here's the preload attribute in action:

```
<video src="myvideo.ogv" preload></video>
```

The *loop* attribute tells the browser to restart the video immediately when it's finished playing, as shown here:

```
<video src="myvideo.ogv" loop></video>
```

Finally, the *autoplay* attribute makes the video automatically play when the page is loaded. For most purposes, this is generally a bad idea from a usability standpoint. Most users will want control over the video; they'll play it when their attention is focused and they're ready to consume the video element. And even with the *autoplay* attribute enabled, your visitors might have that option disabled in their browsers. For that reason, along with the usability problem, I recommend not using the *autoplay* attribute with one notable exception: if you don't include the *controls* attribute, then you need to include *autoplay*; otherwise, the video won't play and visitors will have no way to start it. Here's an example of the *autoplay* attribute:

```
<video src="myvideo.ogv" autoplay></video>
```

Putting it together, a real-world video element looks like this:

```
<video src="myvideo.ogv" width="320" height="240" controls></video>
```

The preceding examples all work well if your visitors have a browser such as Firefox 3.5 or later or Opera 10.5 or later. However, what if a visitor has Internet Explorer? In that case, you'll need to encode the video so that it can be played in Internet Explorer. The *<video>*

tag enables more than one source (via the source element) which you can capitalize on by including links to multiple versions of a video. You can also add a *type* attribute to tell the browser a bit more about the video file to which you're linking. For example, a <*video*> tag that includes the Ogg container video in the preceding example as well as an H.264 video in an mp4 container and a WebM container video would look like this:

```
<video width="320" height="240" controls>
    <source src="myvideo.mp4" type="video/mp4">
    <source src="myvideo.ogv" type="video/ogg">
    <source src="myvideo.webm" type="video/webm">
</video>
```

Additionally, an optional codec portion of the type attribute can also indicate to the browser which codec the audio and video portions of the file use. The use of the codec option is beyond the scope of this book.

With those two options you now have Internet Explorer 9 and Safari covered (thanks to the mp4 container); Firefox and Chrome covered (thanks to the Ogg container); and other browsers covered too (thanks to the WebM container).

The <embed> Tag: Your Fallback Plan

But what happens when someone visits your site with an older browser that doesn't support HTML5? In this case, they won't be able to view video through the <*video*> tag. Luckily, older browsers will simply ignore the video tag so its mere presence won't cause errors. However, you still need to find a way for those visitors to view the video.

You'll find that most users of Internet Explorer also have Adobe Flash installed. With that in mind, you can also include a Flash version of the video on your page. You can include an extra element with the help of the <*embed*> tag. Adobe Flash can play H.264 encoded video with AAC audio; therefore, you don't need to convert your video to yet another format. Here's an example:

```
<embed src="myvideo.mp4" type="application/x-shockwave-flash"
    width="320" height="240" allowscriptaccess="always"
    allowfullscreen="true">
```

Placing a Video Clip on a Web Page

Now that you've got a handle on the theory, it's time to put it into practice with an exercise.

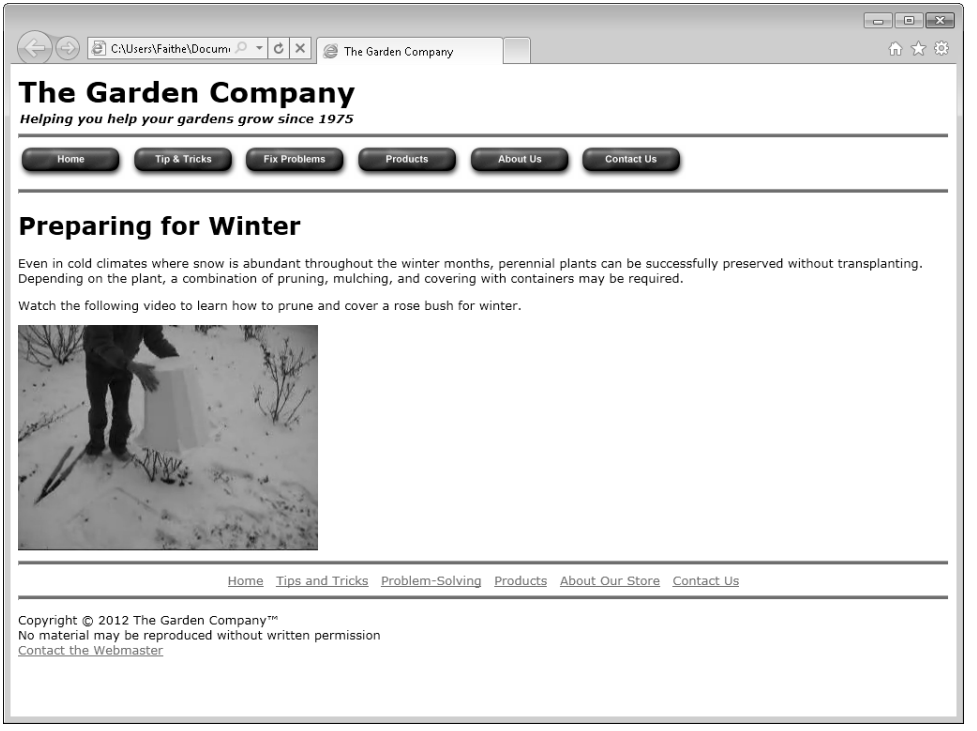
In this exercise, you'll add a video to an HTML page as an embedded clip with the <*video*> tag, and provide an alternative copy as a downloadable link with the <*a*> tag. You'll also practice embedding the clip with the <*embed*> tag.

SET UP Use the *winter.html, myvideo.mp4, myvideo.webm, and myvideo.ogv* files in the practice file folder for this topic. These files are located in the Documents\Microsoft Press\HTML5 SBS\15Video\AddVideo.

1. Open the *winter.html* file in Notepad and in Internet Explorer 9 (or some other HTML5-compliant browser).
2. In the *#main* division, immediately before its closing * * tag, enter the code for inserting video.

```
<p>Watch the following video to learn how to prune and cover a rose bush
for winter.</p>
<video width="320" height="240" autoplay controls>
<source src="myvideo.mp4">
<source src="myvideo.webm">
<source src="myvideo.ogv">
</video>
</div>
```

3. Refresh Internet Explorer to view the clip on the page.



You should see the video and it should start playing automatically. If it doesn't, you might not be using an HTML5-compliant browser.

4. Return to Notepad. Immediately before the closing *</video>* tag, add an *<embed>* tag to play the clip via Flash.

```
<p>Watch the following video to learn how to prune and cover a rose bush
for winter.</p>
<video width="320" height="240" autoplay controls>
<source src="myvideo.mp4">
<source src="myvideo.webm">
<source src="myvideo.ogv">
<embed src="myvideo.mp4" type="application/x-shockwave-flash"
    width="320" height="240" allowscriptaccess="always"
    allowfullscreen="true">

</video>
</div>
```

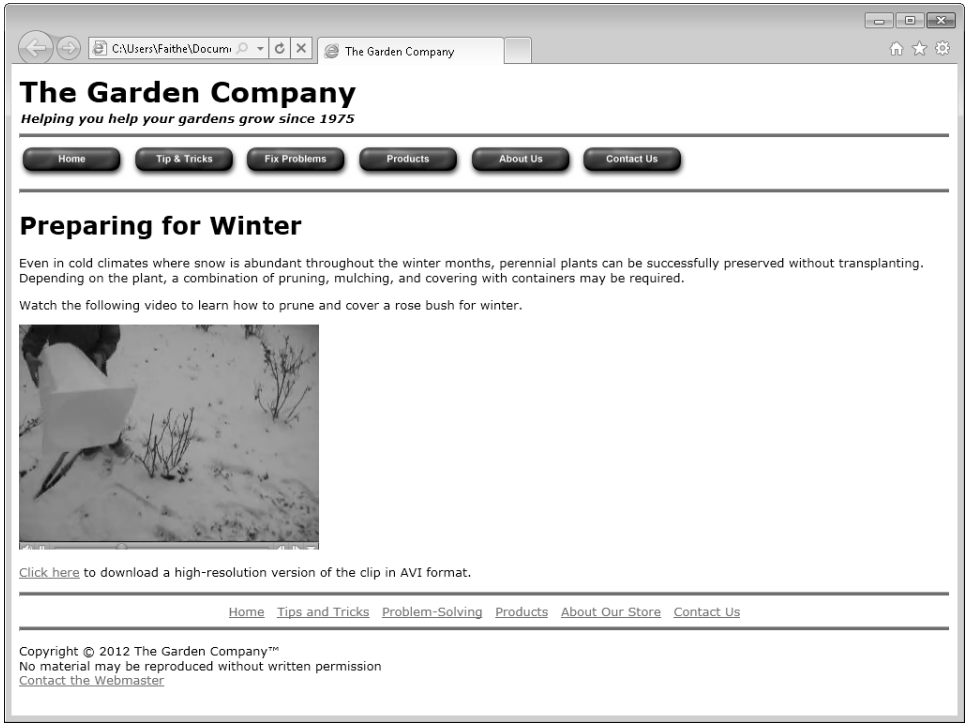
5. Enter the following after the *<embed>* tag:

```
<p>Click here to download a high-resolution version of the clip in AVI
format.</p>
```

6. Make the words *Click here* into a hyperlink that points to the file *myvideo.avi*.

```
<p><a href="myvideo.avi">Click here</a> to download a high-resolution
version of the clip in AVI format.</p>
```

7. Save your work in Notepad, and then refresh the page in Internet Explorer to see the changes.



Note If a security warning appears in the browser window, you might need to click a button to allow the Flash content to play.

CLEAN UP Close the Notepad and Internet Explorer windows.

Incorporating Audio on a Web Page

The good news is that by working your way through the video information in this chapter, you've already learned nearly all the background that you need to play audio on a Web page. The bad news is that the same format and encoding problems that plague video on the Web also apply to audio, except that the audio problems are a bit worse. This section examines the <*audio*> tag and its alternatives.

Playing Audio with the <audio> Tag

You might be thinking that playing audio on a Web page would be easier than video, but for the most part, it's not. You still need to provide for different browsers and encode your audio into different formats. In addition, for the most part, your visitors will still need special plug-ins to play audio. With that said, the <*audio*> tag is new to HTML5 and, assuming that the browser manufacturers can come to some type of agreement (and that's about as possible as me winning the lottery), playing audio on the Web should become easier.

Like the <*video*> tag, the <*audio*> tag supports multiple sources. With no common format, you'll need to encode the audio multiple times to try to get the audio out to the widest possible audience. Also like the <*video*> tag, the <*audio*> tag supports attributes such as *controls*, *autoplay*, *loop*, and *preload*. Therefore, the syntax for the <*audio*> tag is essentially the same as the syntax for the <*video*> tag.

Tip There are numerous applications that convert audio between formats. As with the video conversions, I used VLC to convert the audio when writing this chapter. VLC is available at *http://www.videolan.org/vlc/*.

I've had good success when using the MP3 and Ogg Vorbis formats simultaneously. You'll find support for at least one of these two formats in Firefox, Chrome, Safari, Opera, and Internet Explorer 9. Again, as with video, you'll need to embed your audio stream into a Flash object so older versions of Internet Explorer can use it.

Here's an example that shows the <*audio*> tag with two files, which are called with the help of the <*source*> element that you saw earlier in the video section of this chapter:

```
<audio controls>
    <source src="myaudio.mp3"></source>
    <source src="myaudio.ogg"></source>
</audio>
```

Playing Audio in Older Browsers

As with video, playing audio in older browsers requires the <*embed*> tag. When used with audio, you'll typically use two attributes, *src* and *autostart*; *src* configures the source of the audio, and *autostart* controls whether the audio clip should play automatically upon page load. Adding the <*embed*> tag to the previous example results in this HTML:

```
<audio autoplay loop>
<source src="myaudio.mp3">
<source src="myaudio.ogg">
<embed src="myaudio.mp3">
</audio>
```

By default, content included with <*embed*> will be automatically played. If you don't want this, then add the *autostart="false"* attribute tag, like so:

```
<embed src="myaudio.mp3" autostart="false">
```

Note Even when using <*embed*> to include audio, the visitor must still have software capable of playing the type of file being sent.

One other attribute commonly used with <*embed*> is the *loop* attribute. The *loop* attribute, when set to *true* or *infinite*, restarts the audio clip when it completes. It can also be set to *false* to indicate that the audio clip should play only once. However, the default is to play the audio clip only once; therefore, omitting the *loop* attribute is the same as setting it to *false*.

Placing an Audio Clip on a Web Page

Now you get to practice placing an audio clip. In this exercise, you'll add an audio file to an HTML5 page.

SET UP Use the *index.html*, *myaudio.mp3*, and *myaudio.ogg*.files in the practice file folder for this topic. These files are located in the Documents\Microsoft Press\HTML5 SBS\15Video\AddAudio.

1. Open the *audio.html* file contained in the source code for the book.
2. Immediately above the closing * * tag for the *#main* division, add the codes for the audio clip.

```
<audio autoplay loop>
<source src="myaudio.mp3">
<source src="myaudio.ogg">
</audio>

</div>

</p>
```

2. Before the closing *</audio>* tag, add an *<embed>* tag that will play the clip in a non–HTML5-compliant browser.

```
<audio autoplay loop>
<source src="myaudio.mp3">
<source src="myaudio.ogg">
<embed src="myaudio.mp3">

</audio>

</div>
```

3. Open Internet Explorer 9 or later and view the page.

   The audio should start playing automatically, looping back to the beginning when it completes.

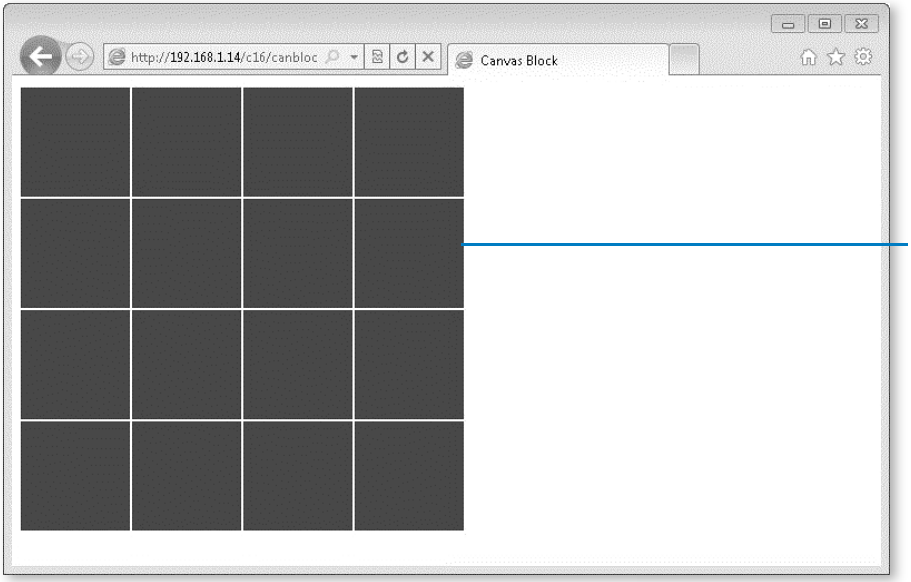
CLEAN UP Close the Notepad and Internet Explorer windows.

Key Points

- Incorporating sound and video is accomplished by providing video and audio files in multiple formats to ensure that your visitors can view the multimedia no matter what browser they're using.
- It's important to understand the different containers and codecs available for video and audio and how those are supported across your visitor's browsers.
- HTML5 introduces the <*video*> and <*audio*> tags, which enable multimedia to be included in Web pages.
- Older browsers don't support the <*audio*> and <*video*> tags, so it's important to provide video in legacy formats such as Flash to enable visitors who use these browswer to view the content as well.
- Use the <*embed*> tag to include audio and video content in a format that non–HTML5-compliant browsers can interpret.

Chapter at a Glance

Add JavaScript code, **page 289**



Use Canvas elements on a page, **page 303**