

HTML5 Training for Web Developers

HTML5 Audio and Video

Lesson 1, Activity 2: Supported Media Types

There has been a lot of discussion about what file formats should be recommended in the HTML5 specification, but it is currently file format neutral. At one point the specification recommended OGG, but it removed that recommendation, in part due to [Apple's refusal to implement OGG](#).

From a practical point of view, to be sure that your media will play in your user's HTML5-compliant browser:

- For audio, you should provide both MP3 and OGG (.ogg) files.
- For video, you should provide both MP4 and OGG (.ogv) files. Also, in May 2010, Google introduced a new file format, [WebM](#), which is meant to address the need for an open, royalty-free, media file format.

Note that you must configure your web server to deliver the audio and video mime types that you use.

- In Apache, you can use `AddType video/ogg .ogv .ogg`. For more information, see https://developer.mozilla.org/en/Properly_Configuring_Server_MIME_Types and scroll down to the section labeled "How to set up your server to send the correct MIME types."
- In IIS, you can add Mime types through Computer Management. For more information, see [http://technet.microsoft.com/en-us/library/cc725608\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc725608(WS.10).aspx).

Lesson 1, Activity 3: The audio Element

Adding audio to an HTML5 page couldn't be more simple. At its most basic, the tag looks like this:

```
<audio src="audio-file.mp3"></audio>
```

However, the above code would not *do* anything or even show up on the page.

To give the user the ability to play and pause the audio, you need to add the `controls` attribute, which can be minimized as shown in the following demo. Open the demo in Chrome, IE9, or Safari to see it work.

Code Sample:

html5-audio-and-video/Demos/audio-controls.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>HTML5 Audio - controls</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<h1>HTML5 Audio - controls</h1>
<article>
<hgroup>
<h2>Casey at the Bat</h2>
<h3>By Ernest Lawrence Thayer</h3>
<h4>from the San Francisco Examiner - June 3, 1888</h4>
</hgroup>
<audio src="../../Media/casey-at-the-bat.mp3" controls></audio>

<p>The Outlook wasn't brilliant for the Mudville nine that day:<br>
The score stood four to two, with but one inning more to play.<br>
And then when Cooney died at first, and Barrows did the same,<br>
A sickly silence fell upon the patrons of the game.</p>
---- C O D E   O M I T T E D ----
```

The controller looks different in different browsers and there is no way to customize the controller using CSS. Opera's controller is shown below:



Audio Formats

If you open the last demo (html5-audio-and-video/Demos/audio-controls.html) in Firefox 3.6, the controller will look like this:



That's because Firefox 3.6 does not support the MP3 format. Opera 10.6 also does not support MP3, but it shows its regular controller; it just won't play the file.

Browsers that support MP3

1. IE (from version 9 beta)
2. Chrome (from version 3.0)
3. Safari 5.0

Browsers that support OGG

1. Firefox 3.5
2. Opera 10.5+
3. Chrome 3.0+

Multiple Sources

Instead of using the `<audio>` tag's `src` attribute, you can nest one or more `<source>` tags within the `<audio>` tag, each with its own `src` attribute and a `type` attribute to let the browser know if the file is of a mime type that it supports (and therefore should bother loading). Browsers will use the first file they support. The code sample below shows how to make our audio tag work in all HTML5-compliant browsers.

Code Sample:

html5-audio-and-video/Demos/audio-multiple-sources.html

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<audio controls>
  <source src="../../Media/casey-at-the-bat.ogg" type="audio/ogg; codecs=vorbis">
  <source src="../../Media/casey-at-the-bat.mp3" type="audio/mpeg">
</audio>
```

```
---- C O D E   O M I T T E D ----
```

audio Tag Attributes

Attribute	Description
src	Points to the audio file. Only used when there are no nested <code>source</code> elements.
controls	Boolean. If present, indicates that controller will be displayed.
preload	Possible values: <ul style="list-style-type: none"> • auto: Browser should choose whether to preload the file. • metadata: Browser should only preload the metadata as the user is likely not to need the file. This is the default. • none: Browser should not preload anything as the user is likely not to need the file.
autoplay	Boolean. If present, audio will begin to play as soon as it has loaded.
loop	Boolean. If present, audio repeats indefinitely.

We have already discussed the `src` and `controls` attributes.

The `preload` attribute is relatively self-explanatory. You should set it to "auto" if you know the browser will need to download the file. Otherwise, you can most likely leave it out or set it to "metadata" (the default).

autoplay

You can get the audio file to play in the background by removing the `controls` attribute and adding the `autoplay` attribute as shown in the following demo.

Code Sample:

html5-audio-and-video/Demos/audio-autoplay.html

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<audio controls autoplay>
  <source src="../Media/casey-at-the-bat.ogg" type="audio/ogg; codecs=vorbis">
  <source src="../Media/casey-at-the-bat.mp3" type="audio/mpeg">
</audio>
---- C O D E   O M I T T E D ----
```

Open the file to hear the poem.

loop

And the `loop` attribute simply makes the audio file restart again when it reaches the end.

Code Sample:

html5-audio-and-video/Demos/audio-loop.html

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<div>
  <p>
    <strong>Add Cheering</strong>: <em><a href="http://creativecommons.org/licenses/sampling+/1.0/">Creative Commons License</a></em><br>
    <small>- downloaded from <a href="http://www.freesound.org/samplesViewSingle.php?id=22952">The Freesound Project</a></small>
  </p>
  <audio controls loop>
    <source src="../Media/cheer.ogg" type="audio/ogg; codecs=vorbis">
    <source src="../Media/cheer.mp3" type="audio/mpeg">
  </audio>
</div>
---- C O D E   O M I T T E D ----
```

Lesson 1, Activity 5: The video Element

The video element is very similar to the `audio` element. Like with audio, it can be as simple as:

```
<video src="video-file.mp4"></video>
```

But it's always a good idea to provide multiple `source` options as different browsers support different video types. As with audio, browsers will use the first file they find that they support.

video Tag Attributes

Attribute	Description
<code>src</code>	Points to the video file. Only used when there are no nested <code>source</code> elements.
<code>controls</code>	Boolean. If present, indicates that controller will be displayed.
<code>preload</code>	Possible values: <ul style="list-style-type: none"> • <code>auto</code>: Browser should choose whether to preload the file. • <code>metadata</code>: Browser should only preload the metadata as the user is likely not to need the file. This is the default. • <code>none</code>: Browser should not preload anything as the user is likely not to need the file.
<code>autoplay</code>	Boolean. If present, video will begin to play as soon as it has loaded.
<code>loop</code>	Boolean. If present, video repeats indefinitely.
<code>height</code>	Height in pixels. You should use the actual height of the video.
<code>width</code>	Width in pixels or percentage. You should use the actual width of the video.

Lesson 1, Activity 7: Video - Multiple Sources

Duration: 10 to 15 minutes.

In this exercise, you will create an HTML5 file from scratch that plays video files.

1. Create a new HTML5 file called [video-multiple-sources.html](#) in the [html5-audio-and-video/Exercises](#) directory.
2. Write the code to include the [justin.mp4](#) (mime type is video/mp4) and [justin.ogv](#) (mime type is video/ogg) files as source options. Both files are located in the [html5-audio-and-video/Media](#) directory.
3. Play around with video attributes such as: controls, autoplay, and loop.

Solution:

[html5-audio-and-video/Solutions/video-multiple-sources.html](#)

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>HTML5 Video - controls</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<h1>HTML5 Video - controls</h1>
<video controls autoplay height="480" width="360">
  <source src="../../Media/justin.mp4" type="video/mp4">
  <source src="../../Media/justin.ogv" type="video/ogg">
</video>
</body>
</html>
```

Lesson 1, Activity 8: **Accessibility**

The HTML5 specification includes a element to provide accessibility features for media delivered using the `audio` and `video` elements, including:

- subtitles
- captions
- descriptions
- chapter information
- metadata

This is still in a bit of flux and browsers don't yet support it, but if you want to learn more, check out <http://dev.w3.org/html5/spec/video.html#the-track-element>.

Lesson 1, Activity 9: Scripting Media Elements

The audio and video elements are accessible and controllable via the DOM. Here are some of the important [methods and properties](#):

Method/Property	Description
currentTime	Read/Write. The current play location in seconds.
duration	Read only. The length of the media file in seconds.
played	Boolean. True if media file has been started.
ended	Boolean. True if media file has been played in full.
autoplay	Boolean. Read/Write.
loop	Boolean. Read/Write.
play()	Plays media from current location (as per <code>currentTime</code>)
pause()	Pauses media

The following demo shows how to use the media API to create rudimentary custom controls and to report the current time of a media file.

Code Sample:

<html5-audio-and-video/Demos/audio-javascript.html>

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<script>
  var playTimer;

  window.addEventListener("load",function() {
    document.getElementById("cmd-play").addEventListener("click",play,false);
    document.getElementById("cmd-pause").addEventListener("click",pause,false);
    document.getElementById("cmd-restart").addEventListener("click",restart,false);
  },false);

  function play() {
    document.getElementById("my-audio").play();
    reportTime();
  }

  function reportTime() {
    var curTime = document.getElementById("my-audio").currentTime;
    document.getElementById("play-time").innerHTML=Math.round(curTime);
    playTimer = setTimeout(reportTime,500);
  }

  function pause() {
    document.getElementById("my-audio").pause();
    if (typeof playTimer != "undefined") {
      clearTimeout(playTimer);
    }
  }

  function restart() {
    document.getElementById("my-audio").currentTime=0;
    play();
  }
</script>
</head>
<body>
<h1>HTML5 Audio - JavaScript</h1>
<article id="poem">
  <hgroup>
    <h2>Casey at the Bat</h2>
    <h3>By Ernest Lawrence Thayer</h3>
    <h4>from the San Francisco Examiner - June 3, 1888</h4>
  </hgroup>
  <audio id="my-audio">
    <source src="../Media/casey-at-the-bat.ogg" type="audio/ogg; codecs=vorbis">
    <source src="../Media/casey-at-the-bat.mp3" type="audio/mpeg">
  </audio>
  <menu type="toolbar">
    <button id="cmd-play" title="Play">Play</button>
    <button id="cmd-pause" title="Pause">Pause</button>
    <button id="cmd-restart" title="Restart">Restart</button>
  </menu>
  <div id="time-display">
    Time: <output id="play-time">0</output> seconds.
  </div>
  ---- C O D E   O M I T T E D ----
```

Notice:





1. The menu with the three buttons.
2. The output element to show the current time.
3. The JavaScript:
 1. When the window loads, we add event listeners for each of the three menu buttons to call the `play()`, `pause()`, and `restart()` functions.
 2. `play()` - calls the audio element's `play()` method and then calls the `reportTime()` function.
 3. `reportTime()` - populates the output element with the audio element's `currentTime` value, rounded to the nearest second. Iteratively calls itself every half second to recheck the `currentTime` value.
 4. `pause()` - calls the audio element's `pause()` method and clears the timer.
 5. `restart()` - sets the audio element's `currentTime` value to 0 and calls `play()`.

Lesson 1, Activity 11: Media API

Duration: 20 to 30 minutes.

In this exercise, you will add a feature to the preceding demo that allows the user to jump to the beginning of a stanza.

1. Open html5-audio-and-video/Exercises/audio-javascript.html in your editor.
2. Notice that each `<p>` tag now has an id of the format `pos-` and a number. This number represents the time (in seconds) at which this stanza begins.
3. You will add JavaScript code to insert working **play** and **pause** images (found in the [Images](#) folder) at the beginning of each stanza, like this:

-   The Outlook wasn't brilliant for the Mudville nine that day:
The score stood four to two, with but one inning more to play.
And then when Cooney died at first, and Barrows did the same,
A sickly silence fell upon the patrons of the game.
-   A straggling few got up to go in deep despair. The rest
Clung to that hope which springs eternal in the human breast;
They thought, if only Casey could get but a whack at that -
We'd put up even money, now, with Casey at the bat.

- When the **play** image is clicked, the audio should jump to that stanza and play.
- When the **pause** image is clicked, the audio should pause.

Challenge

Add code to the `reportTime()` function so that the stanza currently being played is given the "highlight" class (already defined in [style.css](#)). Make sure to remove the class when the stanza is no longer being played.

Solution:

html5-audio-and-video/Solutions/audio-javascript.html

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<script>
  var playTimer;

  window.addEventListener("load",function() {
    document.getElementById("cmd-play").addEventListener("click",play,false);
    document.getElementById("cmd-pause").addEventListener("click",pause,false);
    document.getElementById("cmd-restart").addEventListener("click",restart,false);
    var stanzas = document.getElementById("poem").getElementsByTagName("p");
    for (var i=0; i<stanzas.length; ++i) {
      insertPlayButton(stanzas[i]);
    }
  },false);

  function play() {
    document.getElementById("my-audio").play();
    reportTime();
  }

  function reportTime() {
    var curTime = document.getElementById("my-audio").currentTime;
    document.getElementById("play-time").innerHTML=Math.round(curTime);
    playTimer = setTimeout(reportTime,500);
  }

  function pause() {
    document.getElementById("my-audio").pause();
    if (typeof playTimer != "undefined") {
      clearTimeout(playTimer);
    }
  }

  function restart() {
    document.getElementById("my-audio").currentTime=0;
    play();
  }

  function insertPlayButton(stanza) {
    var pos=stanza.id.split("-")[1];
    var startHTML = stanza.innerHTML;
    var buttonHTML = "<img src='Images/play.gif' title='Play' onclick='jumpTo(" + pos + ")'><img src='Images/pause.gif' title='Pause' onclick='pause'";
    stanza.innerHTML=buttonHTML+startHTML;
  }

  function jumpTo(pos) {
    document.getElementById("my-audio").currentTime=pos;
    play();
  }
</script>
---- C O D E   O M I T T E D ----
```

Challenge Solution:

html5-audio-and-video/Solutions/audio-javascript-challenge.html

```

<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----

function reportTime() {
    //get the current time from the audio element
    var curTime = document.getElementById("my-audio").currentTime;
    //display the current time (rounded) in the "play-time" element
    document.getElementById("play-time").innerHTML=Math.round(curTime);
    //update the current time in 1/2 second (500 ms). This is done by having the function call itself.
    playTimer = setTimeout(reportTime,500);
    //get all the stanzas in the poem (each stanza is a paragraph)
    var stanzas = document.getElementById("poem").getElementsByTagName("p");
    var pos;
    var stanzaFound=false;
    //loop all the stanzas (all the 'p' tags)
    for (var i=stanzas.length-1; i>=0; --i) {
        //set the stanza's position in the loop's iteration
        pos=stanzas[i].id.split("-")[1];
        //if the position is less than or equal to the current time and the stanza has not yet been found, then...
        if (pos <= curTime && !stanzaFound) {
            //first, set stanzaFound to true
            stanzaFound=true;
            //then highlight the stanza by adding the css class "highlight", which is defined in style.css
            stanzas[i].className="highlight";
        } else {
            //else make sure the the stanza is not highlighted
            stanzas[i].className="";
        }
    }
}
---- C O D E   O M I T T E D ----

```

Lesson 1, Activity 12: Dealing with Non-Supporting Browsers

The audio and video elements have been implemented to hide and ignore any unknown children, which means that we can throw code inside of them to be consumed/displayed by non-HTML5 browsers.

Code Sample:

html5-audio-and-video/Demos/video-non-supporting-browsers.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>HTML5 Video - controls</title>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<h1>HTML5 Video - controls</h1>
<video controls autoplay height="480" width="360">
  <source src="../../Media/justin.mp4" type="video/mp4">
  <source src="../../Media/justin.ogv" type="video/ogg">
  <p class="warning">Your browser doesn't support the video tag.</p>
</video>
</body>
</html>
```

If you don't have any non-HTML5-compliant browsers installed, you may want to install [JETester](#), which is a free tool for seeing how older versions of Internet Explorer will display your pages.

Graceful Degradation

Because the code encapsulated within the HTML5 <video> and <audio> tags is not displayed, we can use that to gracefully degrade to Flash and then from there to an image.

Kroc Camen wrote a "chunk of HTML code" called [Video for Everybody](#), which is freely reusable. The code is shown below (with the WebM format added in):

```
<!-- first try HTML5 playback: if serving as XML, expand 'controls' to 'controls="controls"' and autoplay likewise -->
<!-- warning: playback does not work on iPad/iPhone if you include the poster attribute! fixed in iOS4.0 -->
<video width="640" height="360" controls>
  <!-- MP4 must be first for iPad! -->
  <source src="__VIDEO__.MP4" type="video/mp4" /><!-- WebKit video -->
  <source src="__VIDEO__.webm" type="video/webm" /><!-- WebM Format -->
  <source src="__VIDEO__.OGV" type="video/ogg" /><!-- Firefox / Opera -->
  <!-- fallback to Flash: -->
  <object width="640" height="360" type="application/x-shockwave-flash" data="__FLASH__.SWF">
    <!-- Firefox uses the 'data' attribute above, IE/Safari uses the param below -->
    <param name="movie" value="__FLASH__.SWF" />
    <param name="flashvars" value="controlbar=over&image=__POSTER__.JPG&file=__VIDEO__.MP4" />
    <!-- fallback image. note the title field below, put the title of the video there -->
    
  </object>
</video>
<!-- you *must* offer a download link as they may be able to play the file locally. customise this bit all you want -->
<p>
  <strong>Download Video:</strong>
  Closed Format: <a href="__VIDEO__.MP4">"MP4"</a>
  WebM Format: <a href="__VIDEO__.webm">"WebM"</a>
  Open Format: <a href="__VIDEO__.OGV">"Ogg"</a>
</p>
```

Lesson 1, Activity 13: Getting, Creating and Converting Audio and Video Files

Audio

1. [The Freesound Project](#) - a website for downloading free sounds under the [Creative Commons](#) license. Requires registration.
2. [Audacity](#) - free, open source software for recording and editing sounds. Available for Windows, Mac, and Linux.
3. [oggdropXPd](#) - Windows-based drag and drop tool for quickly converting MP3 files to OGG.
4. [Switch Audio File Converter Software](#) - tool for quickly converting between audio file types. Available for Mac and Windows. Free and commercial licenses available.

Video

1. [Miro Video Converter](#) - free, open source tool for quickly converting between video file types. Available for Mac and Windows.
2. [Windows Live Movie Maker](#) - free tool for making movies (WMV files) on Windows 7.