HTML5 for Audio Applications

HTML5 lets you play music through compliant browsers—no "cloud" required.

PAUL FREITAS

Recently, "cloud"-based music services, from big names like Amazon, Google and Apple, have been getting attention in the press. These services allow you to store your music on a corporate server and access it through your own Internet-connected device anytime you like. It's easy to see the appeal of these services. This is the kind of thing the Internet is for, right?

If you're reading this article, you're probably a Linux user, and as often happens, support for Linux is being neglected by these big corporate solutions. For example, Apple's service relies on its proprietary iTunes application, which doesn't exist in Linux. Other products have a Web interface, but uploading works only through a proprietary "app" not available for Linux users. Who wants to use proprietary software anyway? File-type support is limited as well with all the corporate products I've mentioned. Most of my own music is stored in Ogg Vorbis files, and none of the big company services seem to support it, lack of patents notwithstanding. There also are financial costs associated with the corporate offerings (explicit fees comparable to Internet-hosting costs, vendor lock-in and so on) that also are unattractive.

"Cloud" music services have other downsides as well. They are all intended for personal use. What if you want to share music with other people? (I am, of course, talking about legal music sharing involving files with Creative Commons-type free licensing or recorded cover songs with appropriate royalty payments being made to songwriters through licensing agencies like the Harry Fox Agency.) Cloud services can't help you. Also, transfer to the service is one-way. Aside from listening to your music, once you transfer music to the service, you can't download it again easily if something happens to your personal storage. What if you want to use your own storage solution, like your own personal (and appropriately secured) Internet-accessible Web server? What if you live outside the United States, where some cloud services are not yet available?

All these problems make "cloud" solutions more like fog, obscuring the truth that there is another way. Modern HTML5-compliant Web browsers like Chrome, Firefox (Iceweasel to Debian users) and Apple's Safari all support the HTML5 audio element, which can make audio files as easy to handle as image files. Adobe Flash is not necessary. Any Web server can be your own personal "cloud" server that you can use for personal or business use. With some customized HTML and JavaScript, the interface can be anything you want. Let's see how.

All these problems make "cloud" solutions more like fog, obscuring the truth that there is another way.

Playing Music through Your Browser

A typical browser supports multiple audio file types, regardless of the operating system running it. They all support different file types, however, and there are some interesting surprises, the most notable one being that Firefox/Iceweasel doesn't support MP3 files because of patent and licensing issues. That's okay, because it supports the patent-unencumbered Ogg Vorbis format used by many Linux users. So does Google's Chrome, and even Apple's Safari can play Ogg Vorbis with Xiph's QuickTime Components installed (see Resources).

Let's try it. Imagine you have an Ogg Vorbis file named test.ogg in the directory /home/me/music/. You can open it in a Linux-based browser like Chrome or Firefox with the URL file:///home/me/music/test.ogg. Your browser should load the file and start to play it. You'll see a player in the browser tab/window similar to the one shown in Figure 1.

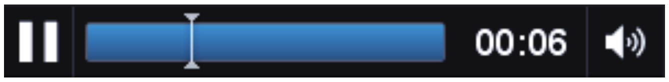


Figure 1. Google Chrome's default audio player control—other browsers have similar control sets, but different appearances.

Browsers render controls differently, but you'll see all the usual controls: play/pause button, position slider, current track position indicator and volume control. Note how much HTML you've had to write so far: none. If all you really want to do is play one file at a time, and you don't care what exactly shows up in the browser window, you can stop reading now.

Most Web users care about appearances, so the default audio controls alone probably won't cut it for most people. Fortunately, you can put an audio player explicitly in any HTML page. It's a lot like adding an image to a page. Instead of using an img element, you use an audio element, and the syntax of the two elements is similar. For example, to load and play test.ogg in an otherwise-empty HTML page using the browser's default controls, you could use the following page:

```
<html>
<body>
    <audio src="test.ogg" controls>Get an HTML5 browser!</audio>
</body>
</html>
```

Note that HTML5 simplifies the syntax of the root HTML element from what you might be used to. The message "Get an HTML5 browser!" will appear only when the page is loaded in a non-HTML5 browser. Other users will see a player element like the one shown in Figure 1, thanks to the controls attribute being set. By default, the audio element has no controls. (HTML5 allows you to set an attribute without a value. If you prefer something more XML-like, you can set an attribute to itself instead—for example, `controls="controls"`.)

It's easy to modify the audio tag for some simple use cases. If you want a sound clip to play in the background with no controls, you can omit the controls attribute:

```
<audio src="test.ogg" autoplay>Get an HTML5 browser!</audio>
```

You can add a loop attribute to make the audio file restart upon completion:

```
<audio src="test.ogg" autoplay loop>Get an HTML5 browser!</audio>
```

As I mentioned earlier, different browsers support different file formats, and they aren't all the same. Ogg Vorbis works for Chrome, Firefox and Safari (with Xiph's extension), but other browsers need something else, like MP3. At the time of this writing, it seems that all browsers support one or the other, but not necessarily both. What can you do to fix this problem?

HTML5 has another element called source that replaces the src attribute of the audio tag. It goes inside the audio element. Unlike the src attribute, you can have multiple source elements inside an audio tag. A browser will load the file referenced by the first source element it finds that it can actually work with. Let's look at the first example again, this time with source attributes:

```
<audio controls>
        <source src="test.ogg"/>
        <source src="test.mp3"/>
        Get an HTML5 browser!
</audio>
```

A browser will attempt to read and play test.ogg first. If it fails for whatever reason (it can't find the file, it can't play that file type and so on), it simply will move on to test.mp3 and use that instead. Use as many source elements as you like, although in practice, two is usually enough.

Adding a Custom User Interface

These simple examples have their uses, of course. Still, most Web developers rather would see a more-consistent user interface than these examples can provide. Having three different UIs appear on three different browsers can affect the usability of a page dramatically. For a professional site, you'll need a professional-looking page. Fortunately, the audio element has a JavaScript API you can use to control the player in real time. To get the interface you want, write it using standard HTML techniques, and set event handlers to access the audio player via JavaScript's API.

Here's a simple example. The following page displays a single play/pause button instead of the native controls:

```
<html>
<body>
    <audio src="test.ogg" id="player" loop>Get an HTML5 browser!</audio>
    <form id="interface">
        <input type="button" value="Play"
         ➥onclick="PlayPause()" id="playpause"/>
    </form>
    <script type="text/javascript">
    var audioPlayer = document.getElementById("player");

    function PlayPause()
    {
        if (audioPlayer.paused)
        {
            audioPlayer.play();
            document.getElementById("playpause").value = "Pause";
        }
        else
        {
            audioPlayer.pause();
            document.getElementById("playpause").value = "Play";
        }
    }
    </script>
</body>
</html>
```

After the page loads, press the Play button to start playing test.ogg in a loop. When a user presses the button, the browser calls the audio player object's `play()` function to start playing the track. The button label then changes to Pause. You then can press the same button to pause the audio, which calls the audio player object's `pause()` function to pause playback.

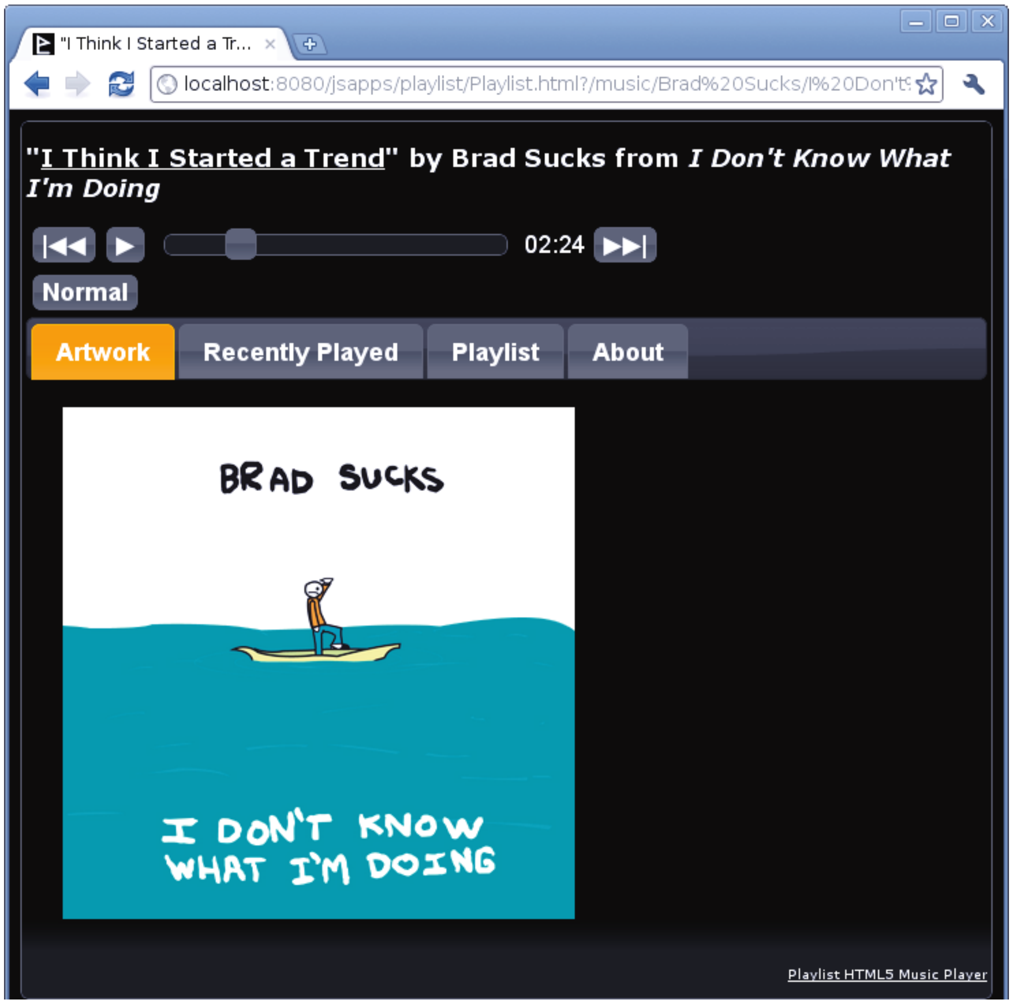
There are many attributes and methods associated with an audio player. You can change the current play time of the player by changing the `currentTime` property. Change the volume by changing the `volume` attribute. There are many others, not all of them implemented in all browsers at the time of this writing. For more information, see the W3C HTML5 specification listed in the Resources section of this article, as well as the documentation for your preferred browser.

A More-Detailed Example: Playlist HTML5 Audio Player

By now you've seen the basics of developing HTML5 for audio applications. You might want to see a more-detailed example. Unfortunately, magazine articles are limited in size, and Web development code takes up a lot of space. Instead, I'd like to refer you to an open-source project I've written called the Playlist HTML5 Audio Player (see Resources). Here is some background on the project.

The goal of Playlist is to provide a user interface for playing a collection of audio files in series, like an album of recorded music. To use the interface, all you need are a collection of audio files and a text file, called a playlist, that lists the files individually. If cover art is available, Playlist will show that as well. Playlist also can load extra information about the collection for display in its About tab. Neither of the last two items are required. Figure 2 shows Playlist playing some freely redistributable sample music.

Playlist supports controls commonly found on a car's CD player, including a shuffle button. It will loop through the


"I Think I Started a Tr...
localhost:8080/jsapps/playlist/Playlist.html?/music/Brad%20Sucks/I%20Don't
02:24
Normal
Artwork
Recently Played
Playlist
About
BRAD SUCKS
I DON'T KNOW WHAT I'M DOING
Playlist HTML5 Music Player

"I Think I Started a Trend" by Brad Sucks from *I Don't Know What I'm Doing*

Figure 2. Playlist HTML5 Audio Player Playing Some Freely Redistributable Sample Music

chosen playlist indefinitely until you pause it. It has jQuery-based controls and tabs, which allow you to change the appearance easily using jQuery's Themeroller tool. Thanks to jQuery's table-sorting and re-ordering abilities, you can re-organize the track order in the Playlist tab any way you like for the duration of the session. For example, Wikipedia says that the Beatles' song "Her Majesty" originally appeared between "Mean Mr. Mustard" and "Polythene Pam" on the *Abbey Road* album. With Playlist, you can drag it from the end of the album and drop it between the other two tracks to hear what that actually sounds like. Playlist is client-side only, so the next time you load your *Abbey Road* page, "Her Majesty" will be back at the end of the album again.

You can use Playlist for your own personal music collection on your own computer through its own Web server. If you want the full "cloud" experience, you also can use it on your own Web site on any of the ubiquitous Web-hosting services that are available today. You don't need any special server-side tools like PHP or MySQL, because Playlist contains no server-side code. You can access your music locally or remotely, through any HTML5-compliant browser that supports your preferred audio file type. I use Playlist for my own music collection, and for my purposes, it works as well as or better than Linux players like Rhythmbox, Exaile and Amarok.

Playlist also can be useful for more-commercial applications. Bands, musicians and songwriters often want to distribute recordings of their own work as part of their overall promotional strategy. If the recording is of original music, something like Playlist can be used directly. If any of the songs are re-recordings of other songwriters' work, appropriate licenses must be obtained, such as the mechanical licenses available for US distribution from the Harry Fox Agency. Accounting for such licenses must happen at least in part on the server, but you still can use Playlist for the UI.

Installing Playlist is reasonably straightforward. In its most simple form, Playlist HTML5 Audio Player consists of a series of ordinary Web files, including Playlist.html (an HTML5 file), Playlist.js (a JavaScript file) and a series of accessory files, including images and jQuery theming elements. All these files are contained in a directory named jsapps (think "JavaScript Applications"). Drop this folder on a Web server where it can be accessed easily. The content to be played (which is, of course, separated from the player) goes somewhere else that makes logical sense to your site, like a folder named music at the same level as jsapps.

Once your audio files are in place, you will need a text file in the same directory named Playlist.m3u that lists the files to play. For example, if you have two tracks named Track1.ogg and Track2.ogg, your file would look like this:

```
Track1.ogg
Track2.ogg
```

You can use command-line tools like `ls` or GUI tools like Audio Tag Tool to create Playlist.m3u more easily.

Finally, you need a file that redirects the browser to Playlist.html in the jsapps directory, along with query parameters that reference the Playlist file. For your convenience, there's one named redirect.html in the jsapps/playlist directory that will do the job via JavaScript. Copy or link that file into your music's directory, alongside Playlist.m3u. I often name the redirect file index.html. Add cover art (Cover.jpg) and HTML-formatted additional information (About.xml) if you like. You're done. Browse to the redirect file and see what happens.

I have Playlist set up on my own Web site so you can easily try it (see Resources). To try Playlist on your own machine, download it from the project home on SourceForge (see Resources). I've packaged it inside a version of Jetty, an open-source Java-based Web server that will run on any platform with a Java Runtime Environment. I've added Jetty as a convenience, but you don't need to use it. You either can download the source from the project page with Git or get the full download with Jetty and copy the music and jsapps directories out of Jetty's webapps directory into an appropriate location on your own Web server. If you want to use Jetty, run `start.sh` (or `start.bat` in Windows) to get the server going, then browse to it via port 8080 (for example, http://localhost:8080/music). All of these test cases will cause freely redistributable sample music to play that will give you an idea of Playlist's capabilities.

Conclusion (and Warning)

HTML5 is a powerful tool for listening to audio via the Internet, as powerful as anything the "cloud" services have to offer and much more versatile. Coding audio into HTML5 pages is fairly straightforward. If writing HTML doesn't interest you and you're looking for a packaged solution, try my open-sourced Playlist HTML5 Audio Player.

I'll end with a warning. With HTML5's great power comes great responsibility—to yourself first and foremost. We've all heard about lawsuits and arrests related to copyright violations and file sharing. You don't want to be part of that.

To protect yourself, use HTML5 audio responsibly. Don't put nonfree music in a publicly accessible or search-engine-crawlable location on any Web server. Firewalls, SSL certificates, Web server configuration files (including Apache's htaccess and norobots.txt), user authentication and other common-sense server-side strategies can help you enjoy your audio anywhere while keeping your private music collection private. If you're distributing music legally, make sure you have all appropriate rights and licenses, mechanical and otherwise.■

Paul Freitas (paulfreitas.com) is an independent technology consultant based in Boise, Idaho.

Resources

HTML5 in General: **http://www.w3.org/TR/html5**

HTML5's Audio Element in Detail: **http://www.w3.org/TR/html5/the-iframe-element.html#the-audio-element**

Xiph.org's QuickTime Components: **http://www.xiph.org/quicktime**

Playlist HTML5 Audio Player: **https://sourceforge.net/projects/playlistplayer**

JQuery: **http://jquery.com**

An example of Playlist, served from my own Web site and playing music with a free license: **http://paulfreitas.com/jsapps/playlist/Playlist.html?/music/Brad%20Sucks/I%20Don't%20Know%20What%20I'm%20Doing/Playlist.m3u**

Harry Fox Agency's Web Site (for obtaining "cover music" mechanical licenses): **http://www.harryfox.com**