Web Audio FAQ

By Boris Smus

Boris is a contributor to Web**Fundamentals**

Over the past few months, the WebKit Web Audio API has emerged as a compelling platform for games and audio applications on the web. As developers familiarize themselves with it, I hear similar questions creep up repeatedly. This quick update is an attempt to address some of the more frequently asked questions to make your experience with the Web Audio API more pleasant.

Q: Halp, I can't make sounds!

A: If you're new to the Web Audio API, take a look at the getting started tutorial, or Eric's recipe for playing audio based on user interaction.

Q. How many Audio Contexts should I have?

A: Generally, you should include one `AudioContext` per page, and a single audio context can support many nodes connected to it. Though you may include multiple AudioContexts on a single page, this can lead to a performance hit.

Q: I've got an AudioBufferSourceNode, that I just played back with `noteOn()`, and I want to play it again, but `noteOn()` doesn't do anything! Help!

A: Once a source node has finished playing back, it can't play back more. To play back the underlying buffer again, you should create a new `AudioBufferSourceNode` and call `noteOn()`.

Though re-creating the source node may feel inefficient, source nodes are heavily optimized for this pattern. Plus, if you keep a handle to the AudioBuffer, you don't need to make another request to the asset to play the same sound again. If you find yourself needing to repeat this pattern, encapsulate playback with a simple helper function like `playSound(buffer)`.

Q: When playing back a sound, why do you need to make a new source node every time?

A: The idea of this architecture is to decouple audio asset from playback state. Taking a record player analogy, buffers are analogous to records and sources to play-heads. Because many applications involve multiple versions of the same buffer playing simultaneously, this pattern is essential.

Q: How can I process sound from `audio` and `video` tags?

A: **`MediaElementAudioSourceNode`** is in the works! When available, it will work roughly as follows (adding a filter effect to a sample playing via the audio tag):

```
<audio src="sounds/sample.wav" controls>

var audioElement = document.querySelector('audio');
var mediaSourceNode = context.createMediaElementSource(audioElement);
mediaSourceNode.connect(filter);
filter.connect(context.destination);
```

This feature is tracked in this crbug. Note that in this setup, there is no need to call **`mediaSourceNode.noteOn()`**, the audio tag controls playback.

Q: When can I get sound from a Microphone?

A: The audio input part of this will be implemented as part of WebRTC using **`getUserMedia`**, and be available as a special source node in the Web Audio API. It will work in conjunction with **`createMediaElementSource`**.

Q: How can I check when an `AudioSourceNode` has finished playing?

A: Currently you have to use a JavaScript timer since Web Audio API does not support this functionality. The following snippet from the Getting Started with Web Audio API tutorial is an example of this in action:

```
// Assume source and buffer are previously defined.
source.noteOn(0);
var timer = setTimeout(function() {
  console.log('playback finished');
}, buffer.duration * 1000);
```

There is an open bug to make Web Audio API implement a more accurate callback.

Q: Loading sounds causes the whole UI thread to lock up and my UI becomes unresponsive. Help!**

A: Use the `decodeAudioData` API for asynchronous loading to avoid blocking the main thread. See this example.

Q: Can the Web Audio API be used to process sounds faster than realtime?

A: Yes, a solution is being worked on. Please stay tuned!

Q: I've made an awesome Web Audio API application, but whenever the tab it's running in goes in the background, sounds go all weird!

A: This is probably because you are using `setTimeouts`, which behave differently if the page is backgrounded. In the future the Web Audio API will be able to callback at specific times using the web audio's internal timer (`context.currentTime` attribute). For more information, please see this feature request.

In general, it may be a good idea to stop playback when your app goes into the background. You can detect when a page goes to the background using the Page Visibility API.

Q: How can I change the pitch of a sound using the Web Audio API?

A: Change the `playbackRate` on the source node.

Q: Can I change pitch without changing speed?

A: The Web Audio API could have a PitchNode in the audio context, but this is hard to implement. This is because there is no straightforward pitch shifting algorithm in audio community. Known techniques create artifacts, especially in cases where the pitch shift is large. There are two kinds of approaches to tackle this problem:

- Time domain algorithms, which cause repeated segment echoes artifacts.
- Frequency domain techniques, which cause reverberant sound artifacts.

Though there is no native node for doing these techniques, you can do it in with a `JavaScriptAudioNode`. This code snippet might serve as a starting point.

Q: How can I create an AudioContext at a sample rate of my choosing?

A: Currently there is no support for this, but we're looking into it. See this feature request.

If you have additional questions, feel free to ask them on StackOverflow using the web-audio tag.