Alessandro Fiordelmondo & Nicolò Pretto
alessandro.fiordelmondo@dei.unipd.it
alessandro.fiordelmondo@spes.uniud.it

WEB AUDIO API

03/2022

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

A BRIEF HISTORY

- <*bgsound*> sound in background in a web page
  - Only in Internet Explorer 

- <*embed*>
  - Similar functionality for Netscape
- Flash was the first cross-browser way of playing back audio on the Web
- HTML5 introduce <*audio*> tag
  - Native support for audio playback in all modern browsers
- Web Audio API (2.0 under development)

HTML5 <audio>

```
<audio controls autoplay muted>
  <source src="mysound.ogg" type="audio/ogg">
  <source src="mysound.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

- source
  - Three supported audio formats: MP3, WAV, OGG
- controls

HTML5 <audio>

Attribute	Value	Description
autoplay	autoplay	Specifies that the audio will start playing as soon as it is ready
controls	controls	Specifies that audio controls should be displayed (such as a play/pause button etc)
loop	loop	Specifies that the audio will start over again, every time it is finished
muted	muted	Specifies that the audio output should be muted
preload	auto metadata none	Specifies if and how the author thinks the audio should be loaded when the page loads
src	*URL*	Specifies the URL of the audio file

https://www.w3schools.com/tags/tag_audio.asp

HTML5 <audio>

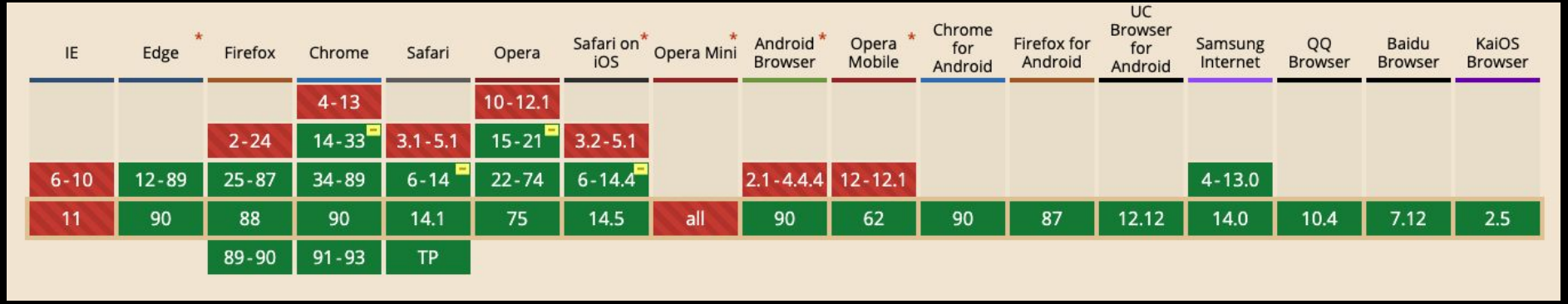
4.0	9.0	3.5	4.0	10.5

- No plug-in
- Limitations:
  - No precise timing controls
  - Very low limit for the number of sounds played at once
  - No way to reliably pre-buffer a sound
  - No ability to apply real-time effects
  - No way to analyze sounds

WAA - WEB AUDIO API

- Completely separated from the <audio> tag
- High-level JavaScript API for processing and synthesizing audio in web applications
- Web Audio API runs in a separate thread, so audio and graphics don't compete as much
- Low-latency
- Events can be scheduled
- Main browsers compatibility

WAA - Compatibility

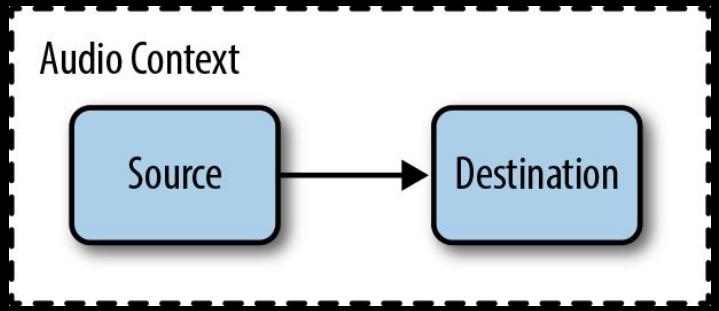


IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
			4-13		10-12.1											
		2-24	14-33	3.1-5.1	15-21	3.2-5.1										
6-10	12-89	25-87	34-89	6-14	22-74	6-14.4		2.1-4.4.4	12-12.1				4-13.0			
11	90	88	90	14.1	75	14.5	all	90	62	90	87	12.12	14.0	10.4	7.12	2.5
		89-90	91-93	TP												

https://caniuse.com/?search=web%20audio%20api
(19/05/21)

WAA - Audio Context

- Audio-processing **graph** built from audio modules linked together, each represented by an **AudioNode**
- Defines how the audio stream flows from its **source** (often an audio file) to its **destination** (often your speakers)
- As audio passes through each node, its properties can be modified or inspected
- The simplest audio context is a connection directly from a source node to a destination node



WAA - Audio Nodes

- **Source nodes**
  - Audio sources for use in the Web Audio API
- **Modification nodes**
  - Process that can be applied to audio sources
  - Splitting and merging audio channels
  - Spatialization and general processing
- **Analysis nodes**
  - Extract time, frequency, and other data from audio
- **Destination nodes**
  - Where to output the audio signal

WAA - Audio Nodes - SOURCE NODE

- **OscillatorNode**
  - Periodic waveform, such as a sine or triangle wave
- **AudioBufferSourceNode**
  - In-memory audio data, stored in an **AudioBuffer**
- **MediaElementAudioSourceNode**
  - HTML5 <audio> or <video> element
- **MediaStreamAudioSourceNode**
  - **MediaStream** (such as a webcam, microphone, or a stream being sent from a remote computer)
  - Multiple audio tracks allowed in the stream

WAA - Audio Nodes - AUDIO PROCESSING NODE

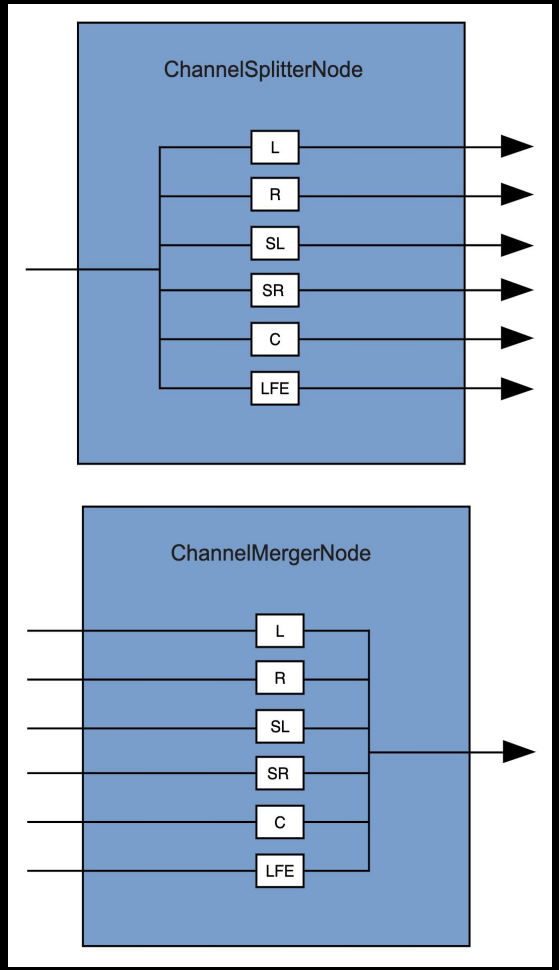
- **BiquadFilterNode**
  - Second order filter
- **ConvolverNode**
  - **Linear Convolution** on a given AudioBuffer
  - Often used to achieve a reverb effect
- **DelayNode**
  - Delay between the arrival of an input data and its propagation to the output
- **GainNode**
  - Change in volume

WAA - Audio Nodes - AUDIO PROCESSING NODE

- **DynamicsCompressorNode**
  - **Compression effect**, which lowers the volume of the loudest parts of the signal in order to help prevent clipping and distortion
- **WaveShaperNode**
  - Non-linear waveshaping distortion
- **PeriodicWave**
  - periodic waveform that can be used to shape the output of an OscillatorNode
- **IIRFilterNode**
  - general infinite impulse response (IIR) filter

WAA - Audio Nodes - SPLITTING AND MERGING

- **ChannelSplitterNode**
  - Separate the different channels of an audio source out into a set of mono outputs
- **ChannelMergerNode**
  - Reunites different mono inputs into a single output



WAA - Audio Nodes - SPATIALIZATION

- **AudioListener**
  - represents the position and orientation of the unique person listening to the audio scene used in audio spatialization
- **PannerNode**
  - represents the position and behavior of an audio source signal in 3D space, allowing you to create complex panning effects
- **StereoPannerNode**
  - Simple stereo panner node that can be used to pan an audio stream left or right

WAA - Audio Nodes - ANALYSIS NODE

- **AnalyserNode**
  - Node able to provide real-time frequency and time-domain analysis information, for the purposes of data analysis and visualization

WAA - Audio Nodes - CUSTOM PROCESSING AUDIO NODE

- Using audio worklets, you can define custom audio nodes written in JavaScript
- **AudioWorklet**
  - available through the AudioContext object's audioWorklet
  - let you add modules to the audio worklet to be executed off the main thread
- **AudioWorkletNode**
  - AudioNode that is embedded into an audio graph and can pass messages to the corresponding AudioWorkletProcessor.
    - Audio processing code running in a AudioWorkletGlobalScope that generates, processes, or analyses audio directly, and can pass messages to the corresponding AudioWorkletNode
  - **AudioWorkletGlobalScope**
    - WorkletGlobalScope-derived object representing a worker context in which an audio processing script is run
    - Generation, processing, and analysis of audio data directly using JavaScript in a worklet thread rather than on the main thread

WAA - Audio Nodes - DESTINATION NODE

- **AudioDestinationNode**
  - End destination of an audio source in a given context — usually the speakers of your device.
- **MediaStreamAudioDestinationNode**
  - The MediaStreamAudioDestinationNode interface represents an audio destination consisting of a WebRTC (Web Real-Time Communication) **MediaStream**

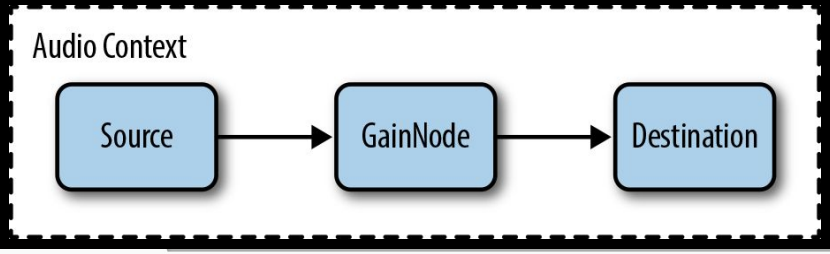
WAA - Audio Nodes - DESTINATION NODE

- **AudioDestinationNode**
  - End destination of an audio source in a given context — usually the speakers of your device.
- **MediaStreamAudioDestinationNode**
  - The MediaStreamAudioDestinationNode interface represents an audio destination consisting of a WebRTC (Web Real-Time Communication) **MediaStream**

WAA - Creating Audio Graph

1. Create the **audio context**
2. Inside the context, create **sources**
3. Create **effects** nodes, such as reverb, biquad filter, panner, or compressor
4. Choose the **final destination** for the audio, such as the user's computer speakers.
5. Establish **connections** from the audio sources through zero or more effects, eventually ending at the chosen destination

WAA - Creating Audio Graph




```
1  // create AudioContext
2  var audioContext = new AudioContext();
3
4  // create the source
5  const source = context.createBufferSource();
6
7  // add AudioBuffer
8  source.buffer = dogBarkingBuffer;
9
10 // add GainNode
11 var gain = context.createGain();
12
13 // Connect source to filter, filter to destination
14 source.connect(gain);
15 gain.connect(context.destination);
```

WAA - Timing Model

- Low-latency precise-timing model
  - Precise timing enables you to schedule events
- Different from the JavaScript timing model used by setTimeout, setInterval, etc.
- Different from the performance clock provided by window.performance.now()
- The absolute times is in **seconds** (not milliseconds!)
- Time is stored as a floating-point value with high precision
- The current time can be retrieved from the audio context via the currentTime property

WAA - Timing Model

- The start() function makes it easy to schedule precise sound playback
- First (when)parameter is in the AudioContext coordinate system (currentTime)
- If the parameter is less than the currentTime, it is played immediately => start(0) plays sound immediately
- To schedule sound in 5 seconds, you would call start(context.currentTime + 5)
- From a specific offset by specifying a second parameter
- Limited to a specific duration with a third optional parameter