Audio Device Client
Better and Faster Audio I/O on Web

Hongchan Choi
Google Chrome
Web Audio API Spec Editor (Audio WG)
Proposal: Audio Device Client

- Work-in-progress! (in Audio CG)
Proposal: Audio Device Client

- Low-level audio I/O
- Better access to hardware
- A dedicated scope runs on RT thread
"But... why?"
"To close the App Gap for audio."
"Web Audio API?"
Issues: Web Audio API (around 2013)

- Extensibility (W3C TAG review, 1st round)
  - No AudioNode subclassing
  - ScriptProcessorNode unfit for purpose
Audio Worklet
Audio Worklet Processor

Audio Worklet Node

User code

index.js
(main thread)

processor.js
(audio thread)

audio callback

user code

audio callback
"How do I port X with it?"
Case study 1: Audio Worklet + WASM
Problems: Audio Worklet + WASM

• Web Audio API render quantum == 128 sample frames (less than 3ms at 44.1Khz)
• Not suitable for large-scale audio application
Case study 2:
AW + SAB + WASM Worker
Worker

SharedArrayBuffer (Audio Data)
  - Input Buffer
  - Output Buffer

AudioWorklet Processor

SharedArrayBuffer (States)
  - Wait/Wake, Buffer indexes

DedicatedWorkerGlobalScope

AudioWorkletGlobalScope
Problems: AW + SAB + WASM Worker

• Low thread priority of Worker
• Overly complex setup for a basic task
Proposal: Audio Device Client

- Low-level audio I/O
  - Isochronous callback-based audio I/O
  - Suitable for WASM-powered audio processing
Proposal: Audio Device Client

- Better access to hardware
  - Configurable render block size, sample rate, and I/O channels
  - Constraints-based hardware configuration
Proposal: Audio Device Client

- A dedicated scope runs on RT thread (if permitted)
  - No more complex plumbing and thread hops
  - For optimum WASM-powered audio processing
Potential Use Cases

- Game audio engine
- Pro-audio applications (music production)
- Client-side spatialization (AR/VR)
- Teleconference
Code Examples
const devices = await navigator.mediaDevices.enumerateDevices();

// Scenario: device #0 and #2 are audio input and output devices respectively.
const constraints = {
  inputDeviceId: devices[0].deviceId,
  outputDeviceId: devices[2].deviceId,
  sampleRate: 32000,
  callbackBufferSize: 512,
  inputChannelCount: 2,
  outputChannelCount: 6,
};
/* async scope: main global scope */
const client =
    await navigator.mediaDevices.getAudioDeviceClient(constraints);
await client.addModule('my-client.js');
client.start();

// when the application ended
client.stop();
import Engine from './my-audio-engine.js';

const process = (input, output) => {
    Engine.process(input, output);
};

setDeviceCallback(process);
Design issues: Integration

- WASM
- AudioContext
- WebRTC
Security/Privacy Concerns

- Real-time priority thread
  - Mitigation: ADC only can be spawned by "top-level document".
- Autoplay policy and secure context by default
github.com/WebAudio/web-audio-cg
(explainer, IDL, code examples, and issue tracker)