WebAudio Modules 2.0, a standard for interoperable WebAudio plug-ins

Michel Buffa (michel.buffa@univ-côtedazur.fr), Shihong Ren (shihong.ren@univ-st-etienne.fr), Steven Yi (stevenyi@gmail.com),

Owen Campbell (owen@ampedstudio.com), Jari Kleimola (jari.kleimola@webaudiomodules.org)

Stéphane Letz (letz@grame.fr), Hugo Mallet (hugo.mallet@53js.fr)

ABSTRACT

In the past, two standards for WebAudio plug-ins existed, with a certain degree of compatibility: WAP (for WebAudio Plugins) and WAM (for WebAudio Modules). Such plugins could be used in different hosts, including a commercial online DAW (AmpedStudio.com), see screenshots at the end of this proposal.

There were some relationships between the two, some authors worked on both projects, and WAMs were a particular case of WAPs, but this was a bit confusing.

All the people involved (Jari Kleimola and Oliver Larkin from WebAudioModules.org, engineers from the online DAW AmpedStudio.com, Michel Buffa and Shihong Ren, Steven Yi from Csound, FAUST DSL team Stéphane Letz, Yann Orlarey, a small french company 53JS.com) decided to merge and unify their work in early 2020.

Now comes WebAudio Modules 2.0 (aka WAM2.0), the unification of previous standards. it comes in the form of a git repo with a SDK, many examples of plugins written in JavaScript, TypeScript, some using the React framework, some written in FAUST, some in CSound.

WAM2.0 supports :

- Modern EcmaScript (modules, dynamic imports, etc.)
- TypeScript (all the API is described in TypeScript interfaces)
- The API has a clear separation of concerns: they are different plugin parts for the GUI, for the DSP part, etc.
- Complete documentation of the API (generated from annotations in the code) + tutorials
- Param automation supported and made easy by the SDK
- Param mapping (you can have internal params that will be interpolated through “exposed params”, for example a mix param can be two gains. If the mix is automated, then internal params will be too).
- Testing framework
- Automatic generation (including GUI) from the online FAUST IDE (write and test WAM2 plugins directly in your web browser)

1. Workshop format

1st Hour

- Introduction to WAM2 (10 minutes)
- First Plugin in JavaScript: Audio Effect (40 minutes)
    - Guided Walkthrough
    - Start with template
    - Walk users through descriptor
    - Implement audio code
    - Add a GUI control and connect to audio code
    - Add a parameter
    - Add presets
- Break/Q&A (10 minutes)

2nd Hour

- Second Plugin in JavaScript: Synthesizer (30 minutes)
    - MIDI Data
    - …
- Plugin Examples (Demo: 15 minutes, perhaps we can pre-record a video for this to ensure it is only 10 minutes)
    - C++
    - Faust
    - Csound
- Break/Q&A (15 minutes)

3rd Hour

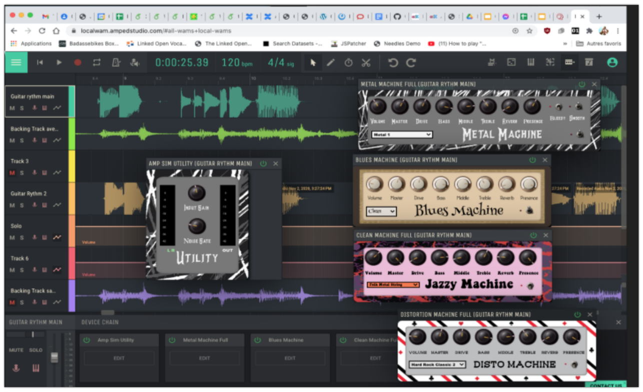
- Coding Time (1 hours)
    - Provide templates and guides for attendees
    - Encourage pair programming or team programming
    - Propose a challenge?

2. Examples

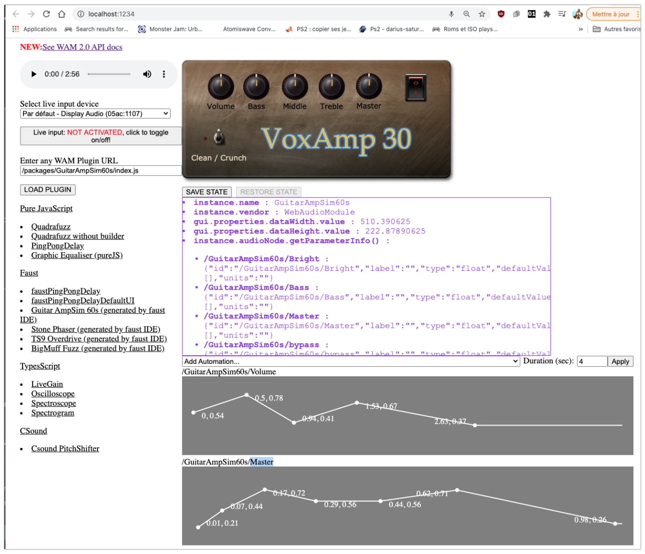
Many examples of hosts, plugins, automation can be tried and are available on our github repo : https://github.com/53js/webaudiomodule

Screenshots:

WAM and WAP plugins in Amped Studio :


0:00:25.39
120 bpm
4/4
METAL MACHINE FULL (GUITAR RYTHM MAIN)
Metal Machine
AMP SIM UTILITY (GUITAR RYTHM MAIN)
Utility
BLUES MACHINE (GUITAR RYTHM MAIN)
Blues Machine
CLEAN MACHINE FULL (GUITAR RYTHM MAIN)
Jazzy Machine
DISTORTION MACHINE FULL (GUITAR RYTHM MAIN)
DISTO MACHINE
Guitar rythm main
Backing Track
Track 3
Guitar Rythm 2
Solo
Track 5
DEVICE CHAIN

The host for testing plugins, from the WAM2.0 github repo: you can dynamically load plugins, perform unit tests on its API implementation, check the metadata it exposes (params, etc.), use it for real (with samples loaded in an audio player or by plugging a microphone or instruments), and you can try automating the params exposed by the plugin:


localhost:1234

NEW:See WAM 2.0 API docs

Select live input device
Par défaut - Display Audio (05ac:1107)

Live input: NOT ACTIVATED, click to toggle on/off!

Enter any WAM Plugin URL
/packages/GuitarAmpSim60s/index.js

LOAD PLUGIN

Pure JavaScript

- Quadrafuzz
- Quadrafuzz without builder
- PingPongDelay
- Graphic Equaliser (pureJS)

Faust

- faustPingPongDelay
- faustPingPongDelayDefaultUI
- Guitar AmpSim 60s (generated by faust IDE)
- Stone Phaser (generated by faust IDE)
- TS9 Overdrive (generated by faust IDE)
- BigMuff Fuzz (generated by faust IDE)

TypesScript

- LiveGain
- Oscilloscope
- Spectroscope
- Spectrogram

CSound

- Csound PitchShifter

```
• instance.name : GuitarAmpSim60s
• instance.vendor : WebAudioModule
• gui.properties.dataWidth.value : 510.390625
• gui.properties.dataHeight.value : 222.87890625
• instance.audioNode.getParameterInfo() :

  • /GuitarAmpSim60s/Bright :
    {"id":"/GuitarAmpSim60s/Bright","label":"","type":"float","defaultVal
    [],"units":""}
  • /GuitarAmpSim60s/Bass :
    {"id":"/GuitarAmpSim60s/Bass","label":"","type":"float","defaultValue
    [],"units":""}
  • /GuitarAmpSim60s/Master :
    {"id":"/GuitarAmpSim60s/Master","label":"","type":"float","defaultVal
    [],"units":""}
  • /GuitarAmpSim60s/bypass :
```

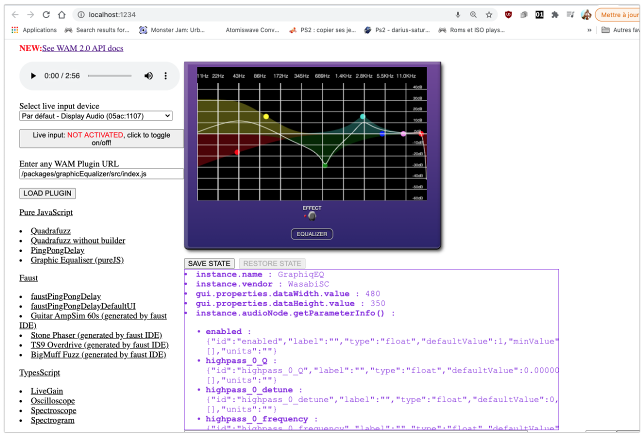
Add Automation... Duration (sec): 4 Apply

/GuitarAmpSim60s/Volume

/GuitarAmpSim60s/Master

Some plugins are written in FAUST and compiled to WebAssembly (the one above has been 100% generated by the Faust Web IDE)

Some are pure JavaScript + WebAudio native nodes::



NEW:See WAM 2.0 API docs

Select live input device

Live input: NOT ACTIVATED, click to toggle on/off!

Enter any WAM Plugin URL

/packages/graphicEqualizer/src/index.js

Pure JavaScript

- Quadrafuzz
- Quadrafuzz without builder
- PingPongDelay
- Graphic Equaliser (pureJS)

Faust

- faustPingPongDelay
- faustPingPongDelayDefaultUI
- Guitar AmpSim 60s (generated by faust IDE)
- Stone Phaser (generated by faust IDE)
- TS9 Overdrive (generated by faust IDE)
- BigMuff Fuzz (generated by faust IDE)

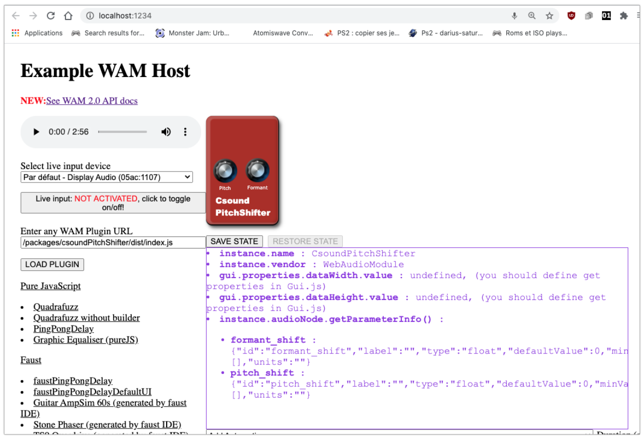
TypesScript

- LiveGain
- Oscilloscope
- Spectroscope
- Spectrogram

```
• instance.name : GraphiqEQ
• instance.vendor : WasabiSC
• gui.properties.dataWidth.value : 480
• gui.properties.dataHeight.value : 350
• instance.audioNode.getParameterInfo() :

  • enabled :
    {"id":"enabled","label":"","type":"float","defaultValue":1,"minValue"
    [],"units":""}
  • highpass_0_Q :
    {"id":"highpass_0_Q","label":"","type":"float","defaultValue":0.00000
    [],"units":""}
  • highpass_0_detune :
    {"id":"highpass_0_detune","label":"","type":"float","defaultValue":0,
    [],"units":""}
  • highpass_0_frequency :
```

Some are in Csound:



Example WAM Host

NEW:See WAM 2.0 API docs

Select live input device

Live input: NOT ACTIVATED, click to toggle on/off!

Enter any WAM Plugin URL

/packages/csoundPitchShifter/dist/index.js

Pure JavaScript

- Quadrafuzz
- Quadrafuzz without builder
- PingPongDelay
- Graphic Equaliser (pureJS)

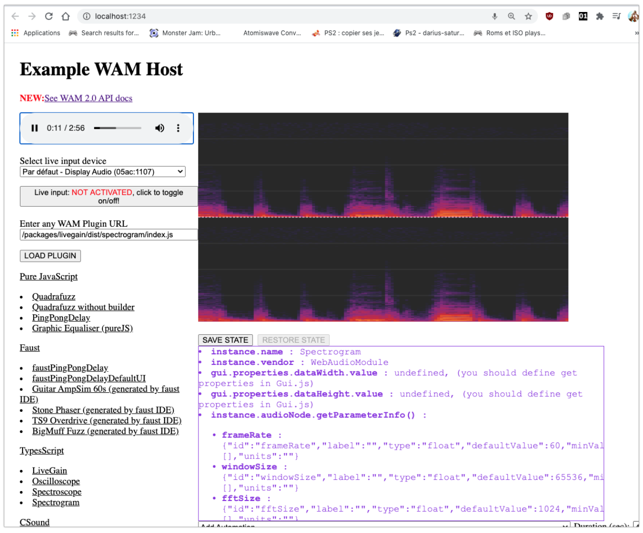
Faust

- faustPingPongDelay
- faustPingPongDelayDefaultUI
- Guitar AmpSim 60s (generated by faust IDE)
- Stone Phaser (generated by faust IDE)

```
• instance.name : CsoundPitchShifter
• instance.vendor : WebAudioModule
• gui.properties.dataWidth.value : undefined, (you should define get
properties in Gui.js)
• gui.properties.dataHeight.value : undefined, (you should define get
properties in Gui.js)
• instance.audioNode.getParameterInfo() :

  • formant_shift :
    {"id":"formant_shift","label":"","type":"float","defaultValue":0,"min
    [],"units":""}
  • pitch_shift :
    {"id":"pitch_shift","label":"","type":"float","defaultValue":0,"minVa
    [],"units":""}
```

Or written in TypeScript::



Example WAM Host

NEW:See WAM 2.0 API docs

Select live input device

Live input: NOT ACTIVATED, click to toggle on/off!

Enter any WAM Plugin URL

/packages/livegain/dist/spectrogram/index.js

Pure JavaScript

- Quadrafuzz
- Quadrafuzz without builder
- PingPongDelay
- Graphic Equaliser (pureJS)

Faust

- faustPingPongDelay
- faustPingPongDelayDefaultUI
- Guitar AmpSim 60s (generated by faust IDE)
- Stone Phaser (generated by faust IDE)
- TS9 Overdrive (generated by faust IDE)
- BigMuff Fuzz (generated by faust IDE)

TypesScript

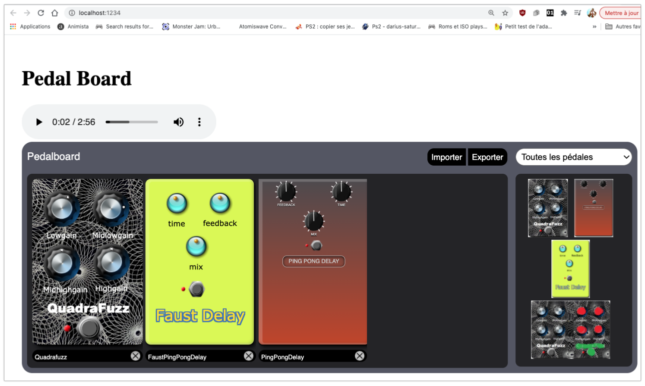
- LiveGain
- Oscilloscope
- Spectroscope
- Spectrogram

CSound

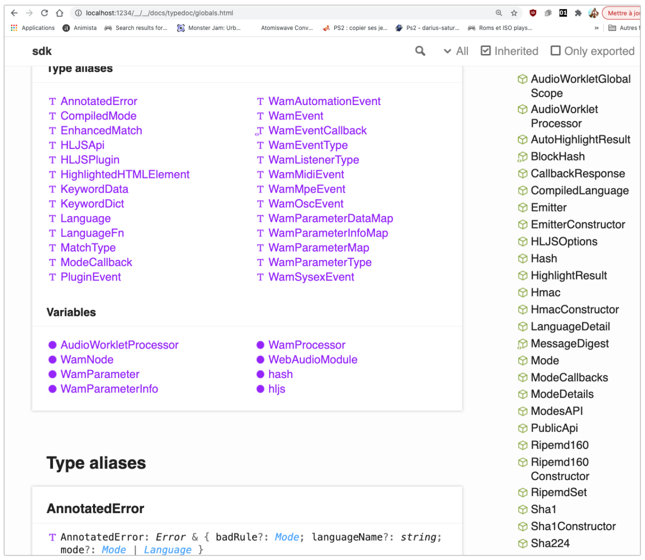
```
• instance.name : Spectrogram
• instance.vendor : WebAudioModule
• gui.properties.dataWidth.value : undefined, (you should define get
properties in Gui.js)
• gui.properties.dataHeight.value : undefined, (you should define get
properties in Gui.js)
• instance.audioNode.getParameterInfo() :

  • frameRate :
    {"id":"frameRate","label":"","type":"float","defaultValue":60,"minVal
    [],"units":""}
  • windowSize :
    {"id":"windowSize","label":"","type":"float","defaultValue":65536,"mi
    [],"units":""}
  • fftSize :
    {"id":"fftSize","label":"","type":"float","defaultValue":1024,"minVal
```

Some plugins are even “meta plugins” (plugins that load and assemble plugins, you see them as “rack plugins”). Below the pedalboard (main container) is a WAM2 plugin:



Documentation:



Type aliases

T AnnotatedError	T WamAutomationEvent
T CompiledMode	T WamEvent
T EnhancedMatch	T WamEventCallback
T HLJSApi	T WamEventType
T HLJSPlugin	T WamListenerType
T HighlightedHTMLElement	T WamMidiEvent
T KeywordData	T WamMpeEvent
T KeywordDict	T WamOscEvent
T Language	T WamParameterDataMap
T LanguageFn	T WamParameterInfoMap
T MatchType	T WamParameterMap
T ModeCallback	T WamParameterType
T PluginEvent	T WamSysexEvent

Variables

● AudioWorkletProcessor	● WamProcessor
● WamNode	● WebAudioModule
● WamParameter	● hash
● WamParameterInfo	● hljs

Type aliases

AnnotatedError

```
T AnnotatedError: Error & { badRule?: Mode; languageName?: string;
  mode?: Mode | Language }
```

- AudioWorkletGlobalScope
- AudioWorkletProcessor
- AutoHighlightResult
- BlockHash
- CallbackResponse
- CompiledLanguage
- Emitter
- EmitterConstructor
- HLJSOptions
- Hash
- HighlightResult
- Hmac
- HmacConstructor
- LanguageDetail
- MessageDigest
- Mode
- ModeCallbacks
- ModeDetails
- ModesAPI
- PublicApi
- Ripemd160
- Ripemd160Constructor
- RipemdSet
- Sha1
- Sha1Constructor
- Sha224