Exploring JavaScript and the Web Audio API

Hugh Zabriskie and Sam Green

Topics

1. Why this API? (5 mins)
2. JavaScript essentials (10 mins)
3. Web Audio API at a high level (10 mins)
4. 5 stages of audio production (20 mins)
5. Sequencer demo! (15 mins)
6. Questions

Setup

Nothing to download! Yay!

Web Audio is built into the JavaScript environment in your browser.

Just open your JavaScript console (Chrome *highly* recommended).

Questions?

Hugh Zabriskie, hzabriskie@college.harvard.edu

Sam Green, samuelgreen@college.harvard.edu

Google! Seriously. Lots of good resources.

Why the Web Audio API?

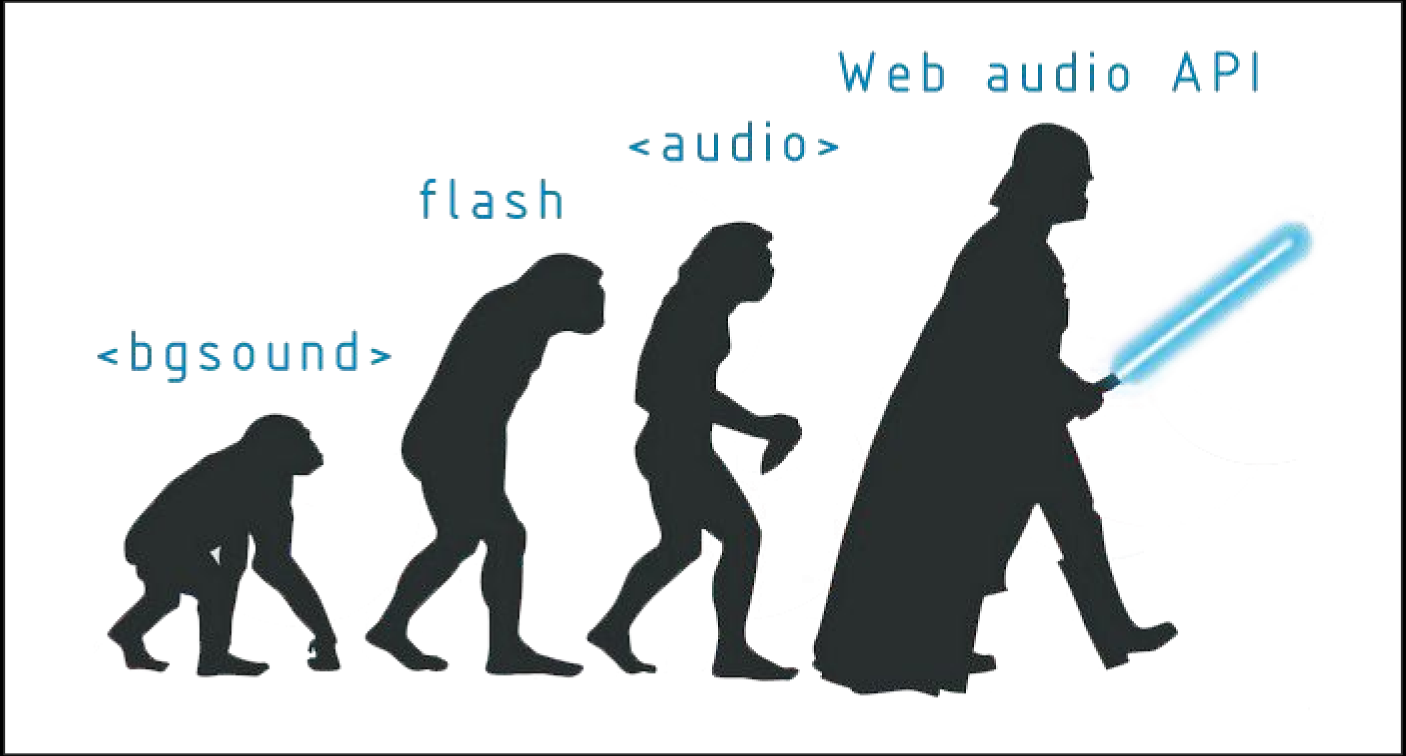


Image src: http://www.machinarium021.com/

Back then...

Sound on the web previously had significant limitations.

- Not cross-browser (no web audio standard)
- No precise timing
- Low limit on simultaneous sounds
- No reliable method for pre-buffering
- No way to apply effects
- No way to analyze sounds

...and now

The Web Audio API offers a standard for working with audio.

- Cross-browser (currently implemented in Chrome, Safari, Opera, Firefox, Edge)
- Precise, low-latency audio
- Support modern game audio engines
- Audio production - mixing, filtering
- Signal processing - raw audio data

JavaScript Essentials

Variables

```
var x = 5;
x = “samuel”;
```

- Dynamic typing (no more “int”)
- Scoping similar to C.
- Global vs. local variables.

Control Flow (1)

```
if (myVar == 1){

} else if (myVar == "1"){

} else {

}
```

Control Flow (2)

```
for (var i = 0 ; i < 100 ; i++){

}
```

```
var i = 0;
while (i < 100){


}
```

Functions

- 2 different syntax styles:

```
function myFunc(){};

var myFunc = function{};
```

- Arguments similar to C
- Functions expect a certain number of arguments, but you *can* call them with fewer (so be careful!)
- Functions are values (whaaaaaat?)

Functions as arguments

- JavaScript is functional, in that functions are values and can be passed to other functions.

Example:

```
var f1 = function(){ console.log(‘hello’);}

function f2(f){f()};

f2(f1);
```

Objects (creation/field access)

- JavaScript supports object-oriented programming.
- Objects wrap some number of values together.
- Remember that functions are values too!
- In JS, calling an object’s “method” is just accessing the function stored in that object.

Generic Object Example

```
var tf = {fn: 'sam',
               ln:'green',
              print:function() {
              console.log(this.fn + '  '+  this.ln);
              }
            };
tf.print();
```

Defined Object Example

```
MyClass = function() {
this.str = 'this is a string';
}
MyClass.prototype.myPrint = function(){
console.log(this.str);
}
var m = new MyClass();
m.print();
```

Asynchronous JavaScript

```
function myFunction(argument, callback){
    // do something
    // wait for something to happen
    // call the callback (perhaps passing back data)
    callback();
}
```

Debugging/ JavaScript Console

- JS console is a feature of modern browsers.
- Useful for debugging your code.
- Also useful for figuring out how to use an API!
- Example of a REPL (read, evaluate, print loop)

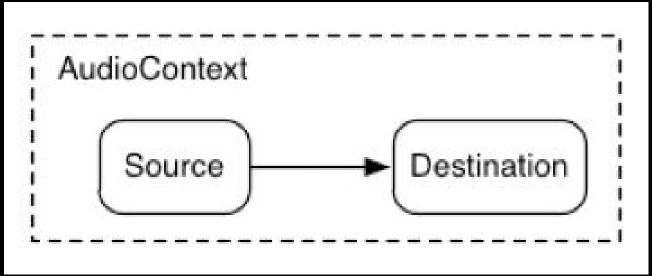
How do I use JavaScript?

- <script> tag at the bottom of an HTML file
  - `<script src=”buffer-loader.js”></script>` essentially copies and pastes the code into the HTML file
  - <script> is for JavaScript :)
- You can also type it into the console
  - `console.log()` will output here
- You don’t have to download anything to use Web Audio
  - It’s built in into your browser!

Web Audio at a high level

Audio as a pipeline

- The source is the audio data that is generated or loaded.
  - Oscillator
  - MP3
  - Microphone
- The destination is where you want to ultimately output the audio data.
  - Laptop speakers (default)
  - ScriptProcessorNode (recording)
- All of the fun stuff happens between these 2 points.
- 5 stages to audio production.



Audio Context

```
context = new AudioContext()
```

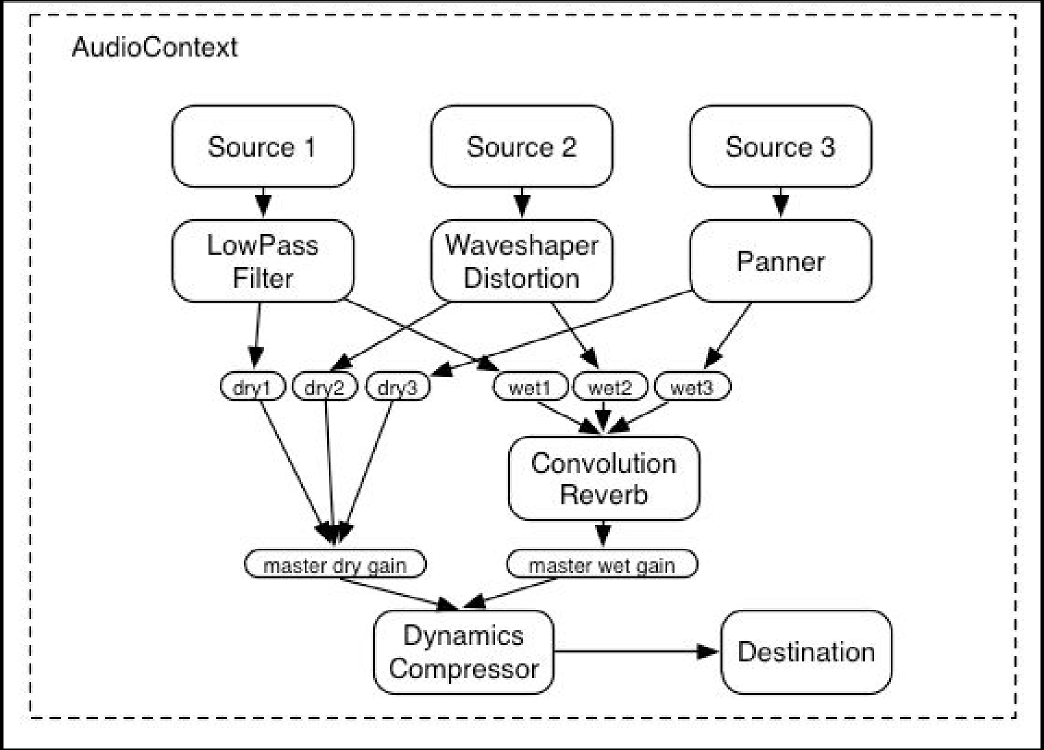
- Global state is maintained through a context
- Should only be created once per session
- Useful properties
    - `destination`: where should the audio play?
    - `currentTime`: precise timestamp for syncing

Assembly as a series of nodes

Everything in the Web Audio API happens as a *node*.

OscillatorNode - generates a tone	`osc = context.createOscillator()`
GainNode - sets volume (gain)	`gain = context.createGain()`
AudioBufferSourceNode - in-memory audio data	`buf = context.CreateBuffer()`
BiquadFilterNode - simple low-pass filter	`bfil = context.CreateBiquadFilter()`
AudioDestinationNode - final destination	`context.destination`
many more…	`context.[autocomplete]`

Example pipeline

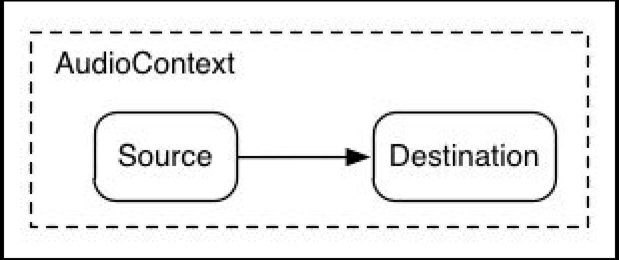


```
var audioCtx = new (window.AudioContext || window.webkitAudioContext)();

//set up the different audio nodes we will use for the app
var analyser = audioCtx.createAnalyser();
var distortion = audioCtx.createWaveShaper();
var gainNode = audioCtx.createGain();
var biquadFilter = audioCtx.createBiquadFilter();
var convolver = audioCtx.createConvolver();

// connect the nodes together

source = audioCtx.createMediaStreamSource(stream);
source.connect(analyser);
analyser.connect(distortion);
distortion.connect(biquadFilter);
biquadFilter.connect(convolver);
convolver.connect(gainNode);
gainNode.connect(audioCtx.destination);

// Manipulate the Biquad filter

biquadFilter.type = "lowshelf";
biquadFilter.frequency.value = 1000;
biquadFilter.gain.value = 25;
```

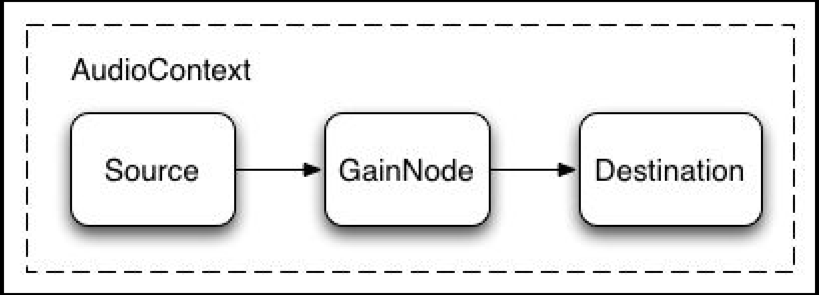
We join pieces of the pipeline (nodes) with the `connect()` method!

The Pipeline of Web Audio Production

Most common pipeline



2nd most common pipeline



1. Source

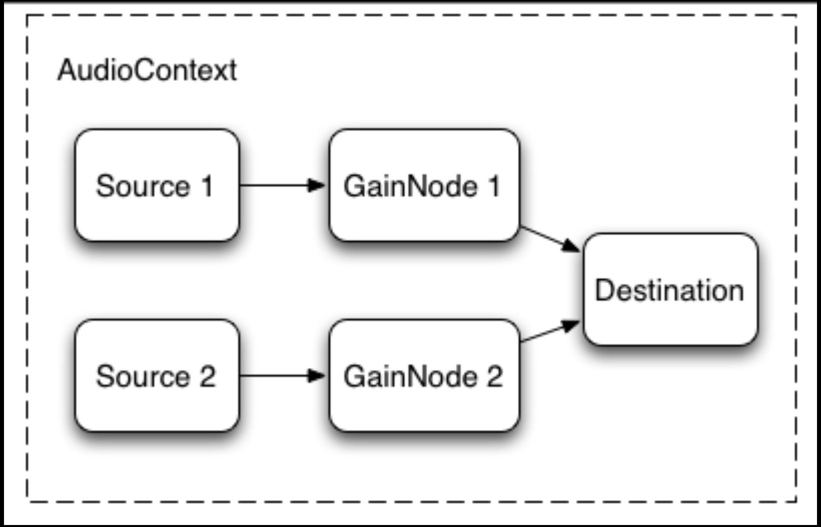
1. Oscillator
    a. Demo time!
2. Buffer load (i.e. MP3)
    a. Uses an XHR request (HTTP request) between client and server
    b. You need to start a basic HTTP server to do this locally!
        i. `python -m SimpleHTTPServer 8080`
    c. Once it loads asynchronously, use the callback function to retrieve the audio data
    d. See buffer-loader.js (http://www.html5rocks.com/en/tutorials/webaudio/intro/js/buffer-loader.js)
        i. Thanks Boris Smus!
3. Microphone
    a. Use `Navigator.getUserMedia()` to request access to the user's microphone

2. Filtering

- *more advanced topic*
- What does a filter do?
    - Filters emphasize or de-emphasize certain parts of the frequency spectrum of a sound
- Examples?
    - Low-pass filter —> makes the sound more muffled
    - High-pass filter —> Makes sounds more tinny
    - Low-shelf filter —> Affects the amount of bass in a sound (like the bass knob on a stereo)
    - Notch filter —> Removes unwanted sounds in a narrow frequency range
- Used to create pink noise, brown noise, white noise, etc.

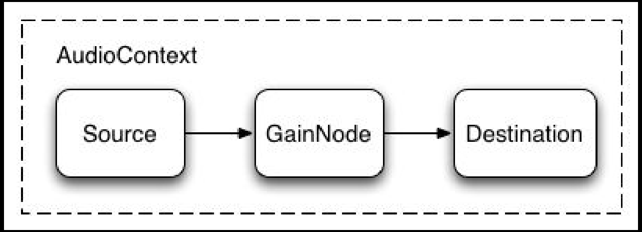
3. Mixing

- *advanced topic*
- AudioPannerNode
  - Pan audio (L/R)
  - Change how audio is distributed
  - 3D effect
- Crossfading
  - Think of a DJ with two tracks - one song is finishing, the DJ fades into the next one
  - Creating multiple pipelines to the same destination


$$\begin{tikzcd}
\text{Source 1} \arrow[r] & \text{GainNode 1} \arrow[dr] & \\
 & & \text{Destination} \\
\text{Source 2} \arrow[r] & \text{GainNode 2} \arrow[ur] &
\end{tikzcd}$$

4. Gain

- Set the volume of a sound.
    - Sets the power of a signal (i.e. amps)



```
context = new AudioContext()
osc = context.createOscillator()
gain = context.createGain()
osc.frequency.value = 440
osc.connect(gain)
gain.gain.value = 0.6 // 0 is no signal, 1 (default) is full signal (loudest setting)
gain.connect(context.destination)
```

DEMO TIME!

5. Output

- Where should the audio go?
  - Doesn’t have to go anywhere necessarily (i.e. pitch detector)
  - Usually it’s the speakers

`context.destination` is an *AudioDestinationNode*

```
context.destination.numberOfOutputs = 0 // nothing leaves here muahaha
```

DEMOS

1. Play my favorite song!
   a. And stop my favorite song :(
2. Music sequencer
   a. http://hughzabriskie.com/sequencer
   b. Username: guest
   c. Password: sequencer

Questions?

Hugh Zabriskie, hzabriskie@college.harvard.edu

Sam Green, samuelgreen@college.harvard.edu

Google! Seriously. Lots of good resources.