
chrome


CAMPUS SEOUL

Musicking on the Web

Workshop Day 1 / Lecture 2

Hongchan Choi

Software Engineer, Google Chrome
Web Audio API + Music Apps

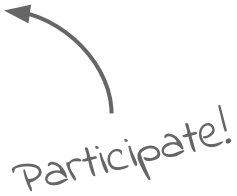
hoch.io

Web Audio API

W3C Editor’s Draft

http://webaudio.github.io/web-audio-api/

https://github.com/WebAudio/web-audio-api/issues

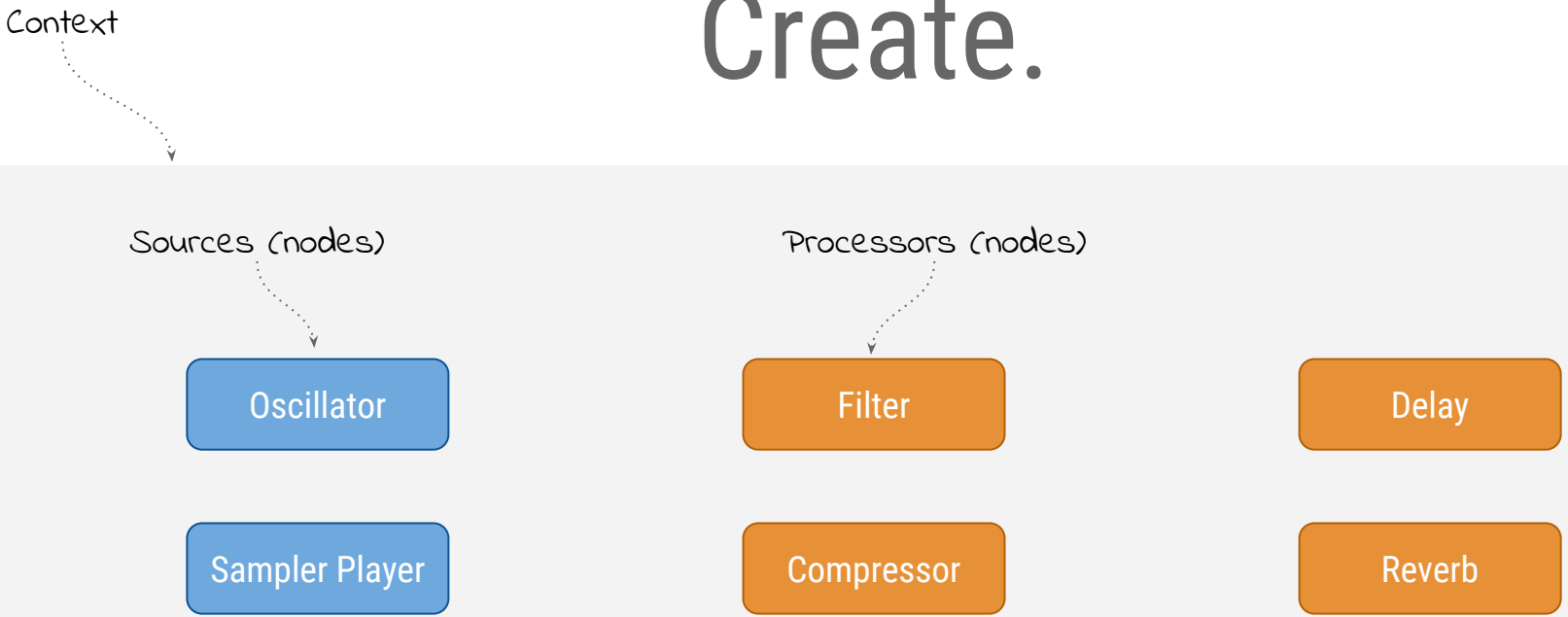


W3C Audio Working Group

Google, Mozilla, Apple, Microsoft, BBC, Dolby and more...

Programming Web Audio API

Create.



SPEAKER

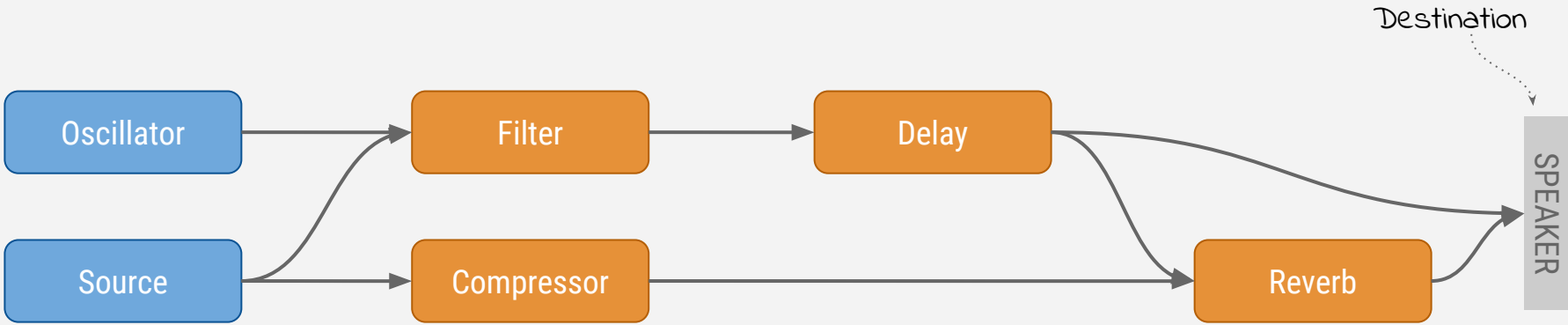
Create.

```
var context = new AudioContext();

var osc = context.createOscillator();
var samp = context.createBufferSource();
var filter = context.createBiquadFilter();
var comp = context.createDynamicsCompressor();

var delay = context.createDelay();
var reverb = context.createConvolver();
```

Connect.


Destination
Oscillator
Filter
Delay
Source
Compressor
Reverb
SPEAKER

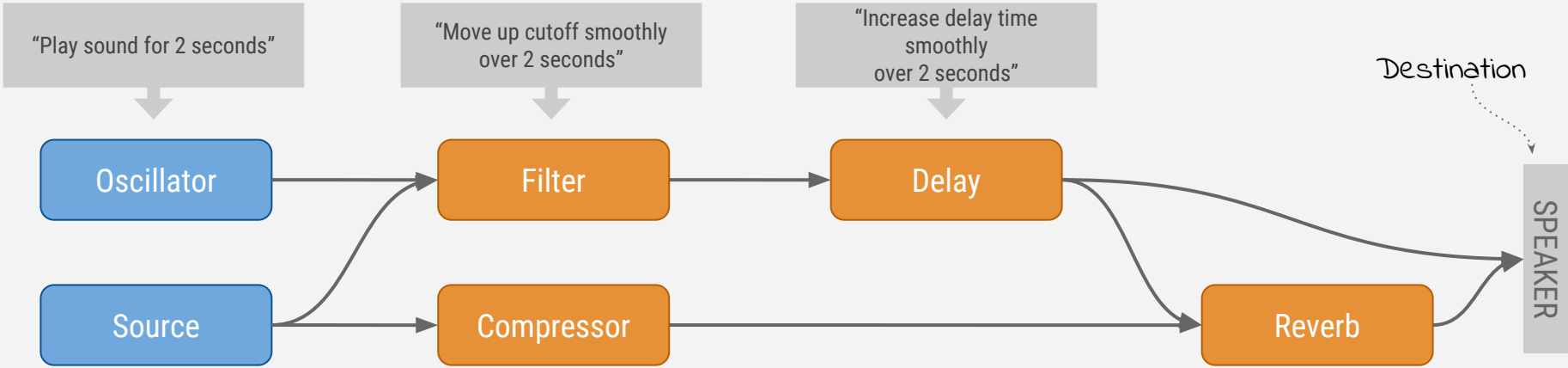
Connect.

```
// cont'd

osc.connect(filter);
filter.connect(delay);
samp.connect(filter);
samp.connect(comp);
comp.connect(reverb);

delay.connect(reverb);
delay.connect(context.destination);
reverb.connect(context.destination);
```

Control.


"Play sound for 2 seconds"
"Move up cutoff smoothly over 2 seconds"
"Increase delay time smoothly over 2 seconds"
Destination
Oscillator
Source
Filter
Compressor
Delay
Reverb
SPEAKER

Control.

```
// cont'd

osc.start(0.0);
osc.stop(2.0);

filter.frequency.setValueAtTime(440, 0.0);
filter.frequency.linearRampToValueAtTime(1000, 2.0);

delay.delayTime.setValueAtTime(0.25, 0.0);
delay.delayTime.linearRampToValueAtTime(1.0, 2.0);
```

Canopy

http://hoch.github.io/canopy


WE NEED TO GO DEEPER...

Primitives

Context, Nodes and Parameters

AudioContext

- For real-time audio synthesis/processing
  - `.currentTime` advances in real time.
  - Runs on top of the browser's audio hardware.

OfflineAudioContext

- For non-realtime synthesis/processing
  - Renders the result into an `AudioBuffer`.
  - Runs (potentially) faster than real time.
- Configure with: Sample rate, Number of channels and Render duration

AudioNode

- Created by `AudioContext` and `OfflineAudioContext`.
- Connects with the other `AudioNode`.
- Has dynamic lifetime.

```
AnalyserNode
AudioBufferSourceNode
BiquadFilterNode
ChannelMergerNode
ChannelSplitterNode
ConvolverNode
DelayNode
GainNode
OscillatorNode
StereoPannerNode
WaveShaperNode
MediaElementAudioSourceNode
MediaStreamAudioDestinationNode
MediaStreamAudioSourceNode
```

AudioParam

- Member of `AudioNode` object.
- Offers automation methods.

```
.setValueAtTime()                    .exponentialRampToValueAtTime()
.linearRampToValueAtTime()           .setTargetAtTime()
```

- Can be connected with `AudioNode` output. (i.e. modulation)

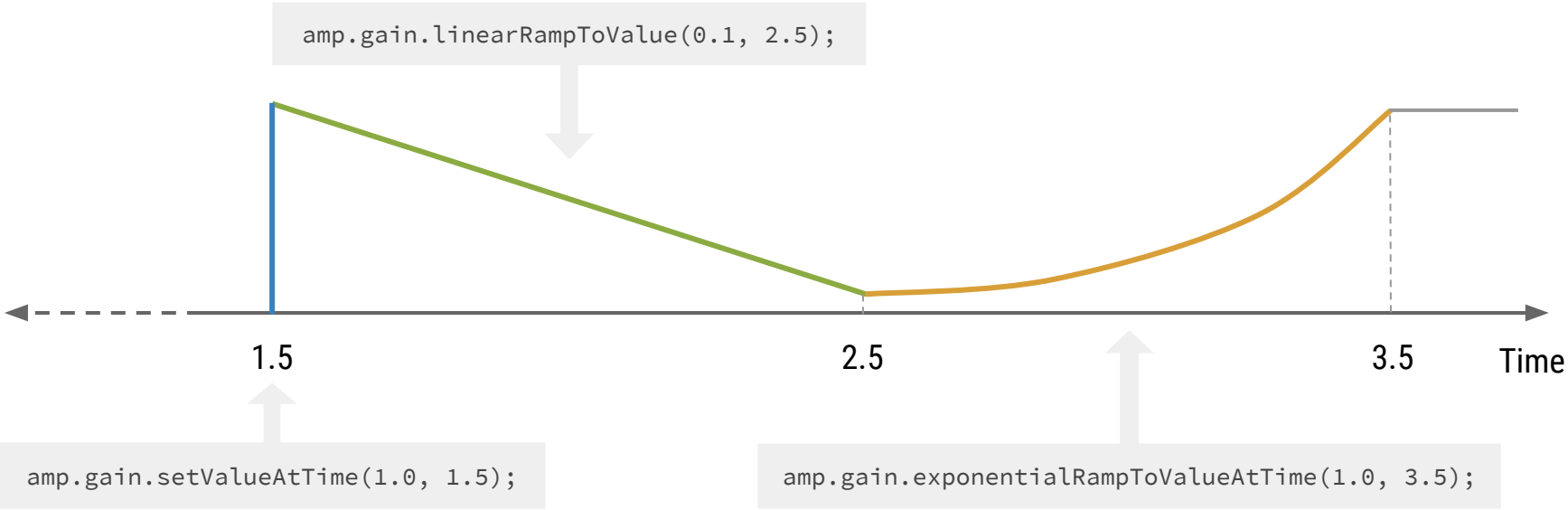
AudioParam: automation

Take a guess what will happen!

```
var amp = context.createGain();


amp.gain.setValueAtTime(1.0, 1.5);
amp.gain.linearRampToValue(0.1, 2.5);
amp.gain.exponentialRampToValueAtTime(1.0, 3.5);
```

AudioParam: automation

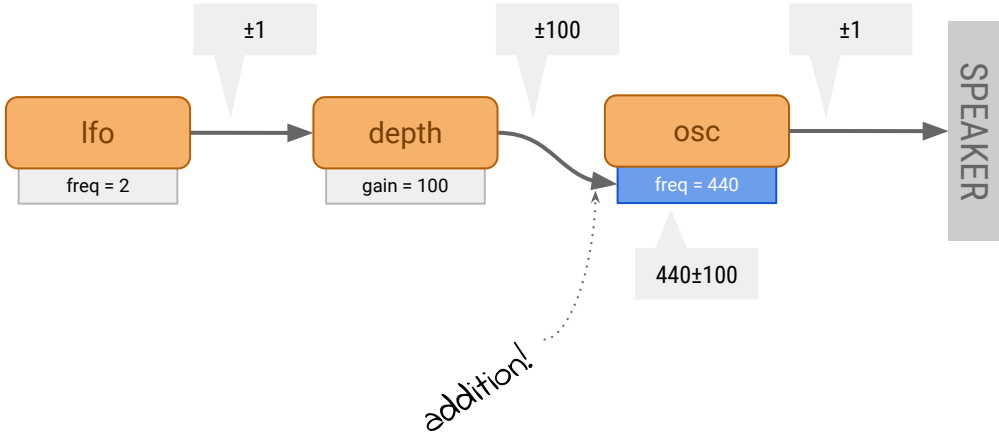


AudioParam: modulation

```
var lfo = myContext.createOscillator();
var depth = myContext.createGain();
var osc = myContext.createOscillator();

lfo.frequency.value = 2;
depth.gain.value = 100;

lfo.connect(depth);
depth.connect(osc.frequency);
osc.connect(myContext.destination);
lfo.start();
osc.start();
```



Moar Canopy

http://hoch.github.io/canopy

Loading Audio File

- **MUST** use `XMLHttpRequest` and `decodeAudioData`.
- **MUST** follow CORS (Cross-origin Resource Sharing) rule.
- This is **asynchronous** process.

Loading Audio File

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'sounds/hello.wav', true);
xhr.responseType = 'arraybuffer';
xhr.onload = function (event) {
      myContext.decodeAudioData(xhr.response, function (buffer) {
            mySource.buffer = buffer;
      });
};
xhr.send();
```

Spiral to the rescue!

https://github.com/hoch/spiral

Spiral is...

- JavaScript library for Web Audio/MIDI API.
- To extend them with the minimum layer of abstraction.

Why Spiral?

- ❏ Non ‘kool-aid’ approach: No need to learn new things.
- ❏ Easy integration with Spiral-Polymer elements.
- ❏ Comes with the boilerplate: Node.js + Bower + Gulp
- ❏ Test framework + Documentation (WIP)

AudioContext.createNodes()

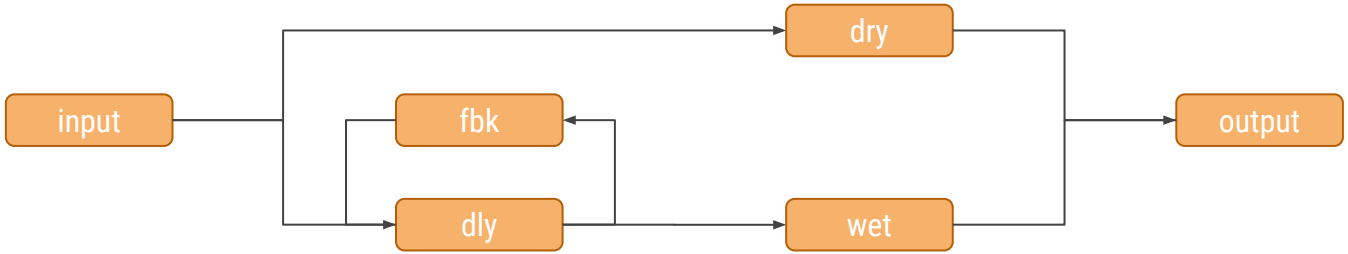
```
function Echo(context) {
  this.input = context.createGain();
  this.wet = context.createGain();
  this.dry = context.createGain();
  this.dly = context.createDelay();
  this.fbk = context.createGain();
  this.output = context.createGain();
}
```

Web Audio API

```
function Echo(context) {
  context.createNodes(this, {
    'input': 'Gain',
    'wet': 'Gain',
    'dry': 'Gain',
    'dly': 'Delay',
    'fbk': 'Gain',
    'output': 'Gain'
  });
}
```

Spiral

AudioNode.to()



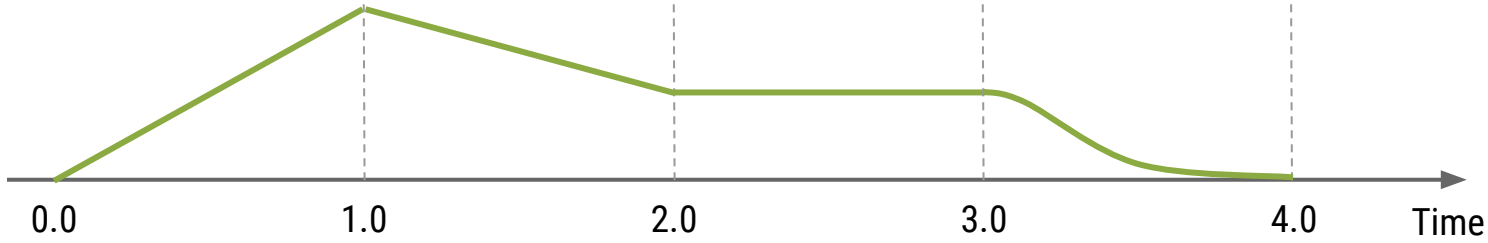

```
this.input.connect(this.dry);
this.input.connect(this.dly);
this.dly.connect(this.fbk);
this.fbk.connect(this.dly);
this.dly.connect(this.wet);
this.dry.connect(this.output);
this.wet.connect(this.output);
```

Web Audio API

```
this.input.to(this.dry, this.dly);
this.dly.to(this.fbk).to(this.dly).to(this.wet);
this.wet.to(this.output);
this.dry.to(this.output);
```

Spiral

AudioParam automations




```
amp.gain.setValueAtTime(0.0, 0.0);
amp.gain.linearRampToValueAtTime(1.0, 1.0);
amp.gain.linearRampToValueAtTime(0.25, 2.0);
var oneSec = 1 / Math.log(1 / 0.001);
amp.gain.setTargetAtTime(0.001, 3.0, oneSec);
```

Web Audio API

```
amp.gain
  .step(0.0, 0.0)
  .line([1.0, 1.0], [0.25, 2.0])
  .slew(0.0, 3.0, 1.0);
```

Spiral

AudioContext.loadAudioFiles()

```
var audioFileData = [
  { name: 'snare', url: 'sound/sd-001.mp3' },
  { name: 'kick', url: 'sound/kd-001.mp3' },
];

function progress(filename) { console.log(filename + ‘ just loaded’); }
function resolve(buffers) { console.log(‘Loading completed successfully.’); }
function reject(buffers) { console.log(‘Loading completed with some errors.’); }

audioContext.loadAudioFiles(audioFileData, onprogress).then(resolve, onreject);
```

And more stuffs...

- ❏ Music math utilities
- ❏ Web MIDI API support
- ❏ Compatibility patching (a.k.a. Monkey patching)

enjoy your lunch!

That’s a wrap!

Workshop Day 1 / Lecture 2