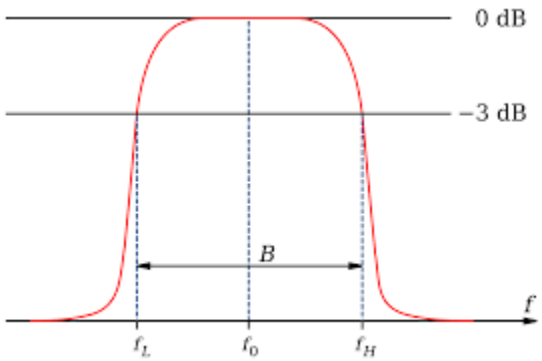
How to setup bandpass filter using webaudio

井民全, Jing, mqjing@gmail.com



Quick

```
function setFilter(filter, type, from, to){
   // filter.type = 'bandpass';
   filter.type = type;
   var geometricMean = Math.sqrt(from * to);
   filter.frequency.value = geometricMean;
   filter.Q.value = geometricMean / (to - from);
}

  var from = 300;
      var to = 30000;
      var filter = this.context.createBiquadFilter();
      setFilter(filter, 'bandpass', from, to);
```

Code

File: client.html

```
<!doctype html>
<html lang="en">
<body>
<h1>Client</h1>
<script src="index.js"></script>
```

File: client.js

```
'use strict';

function setFilter(filter, type, from, to){
   // filter.type = 'bandpass';
   filter.type = type;
   var geometricMean = Math.sqrt(from * to);
   filter.frequency.value = geometricMean;
   filter.Q.value = geometricMean / (to - from);
}

class Test{
   constructor() {
      this.context = null;
      this.buffer = null;
      this.source = null;

      this.startOffset = 0;
      this.startTime = 0;
   }

   initUI(){
      this.addPlayButton();
      this.addPauseButton();
      this.addStopButton();
   }

   play(buffer) {
      this.startTime = this.context.currentTime;
      this.source = this.context.createBufferSource();
      this.source.buffer = this.buffer;

      var from = 300;
      var to = 30000;
      var filter = this.context.createBiquadFilter();
      setFilter(filter, 'bandpass', from, to);

      let gainNode = this.context.createGain();
```

```
      gainNode.gain.value = 3;

      // create filter graph
      this.source.connect(filter);
      filter.connect(gainNode);
      gainNode.connect(this.context.destination);

      // play from begin
      this.source.start(0, this.startOffset % buffer.duration);

      setTimeout(function() {
         alert('set filter')
         var from = 20;
         var to = 200;
         setFilter(filter, 'bandpass', 20, 200);
       }, 5000)
   }

   stop() {
      if (this.source != null) {
         this.source.stop();
         return [1, 'ok'];
      }else{
         let strMsg = '[Error] _uiSoundStop:: this.source == null.';
         return [0, strMsg];
      }
   }

   pause() {
      const [bOk, strMsg] = this.stop();
      if (!bOk)
         return [bOk, strMsg];

      this.startOffset += this.context.currentTime - this.startTime; // Measure how much time
passed since the last pause.
      return [1, 'ok'];
   }


   addPlayButton(){
      this.comm('Play', async () => {
         console.log('this.constructor.name = ', this.constructor.name); // Test
         console.log('clicked');
         this._initAudioContext();
         this.buffer = await
this._loadAudioBuffer("https://archive.org/download/100ClassicalMusicMasterpieces/1685%
20Purcell%20,%20Trumpet%20Tune%20and%20Air.mp3");
         this.play(this.buffer);
```

```
    });
}

addPauseButton(){
    this.comm('Pause', () => {
        let [bOk, strMsg] = this.pause();
        if (!bOk){
            this._showMessageError(strMsg);
        }
    });
}

addStopButton(){
    this.comm('Stop', () => {
        this.stop();
    });
}

comm(strTitle, listener) {
    let button = document.createElement('button');
    button.innerHTML = strTitle;  // set the button content
    button.addEventListener('click', listener);

    let body = document.getElementsByTagName("body")[0];
    body.appendChild(button);
}

_initAudioContext() {
    let contextClass = (window.AudioContext ||
        window.webkitAudioContext ||
        window.mozAudioContext ||
        window.oAudioContext ||
        window.msAudioContext);
    if (contextClass) {
            console.log('Web Audio API is available.')
            this.context = new contextClass();
    } else {
        console.log(' Web Audio API is not available. Ask the user to use a supported browser.')
    }
}

_loadAudioBuffer(url){
    return new Promise(resolve => {
        var request = new XMLHttpRequest();
        request.open('GET', url, true);
        request.responseType = 'arraybuffer';
        // Decode asynchronously
        request.onload = () => {
```

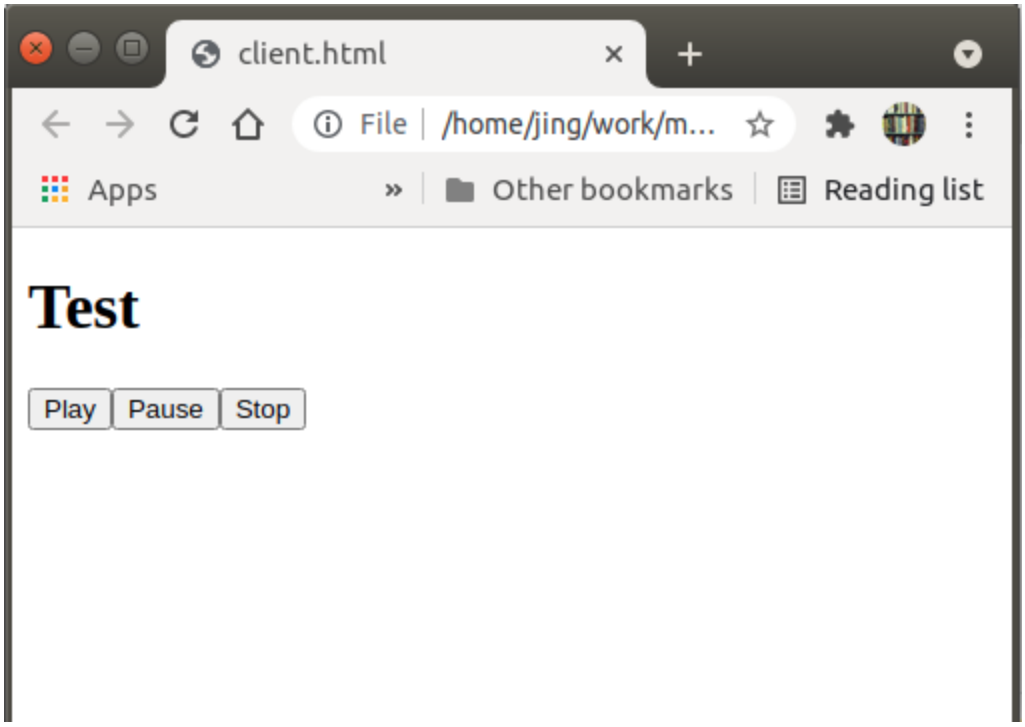
```
            this.context.decodeAudioData(request.response, (theBuffer) => {
                console.log('Get the data.')
                this.buffer = theBuffer;
                resolve(this.buffer);
                console.log('buffer = ', this.buffer);
                }, function () {
                    console.log('[Error] _loadAudioBuffer:: decodeAudioData error.')
                });
            } // end of onload
            request.send();});
        }

    _showMessageError(strMsg) {
        console.log(strMsg);
        alert(strMsg);
    }
} // end of class

let test = new Test();
test.initUI();
```

Run

gio open client.html



References

1. https://developer.mozilla.org/en-US/docs/Web/API/BaseAudioContext/decodeAudioData
2. https://stackoverflow.com/questions/33540440/bandpass-filter-which-frequency-and-q-value-to-represent-frequency-range
3. https://github.com/borismus/webaudioapi.com/blob/master/content/book/Web_Audio_API_Boris_Smus.pdf, page 28.