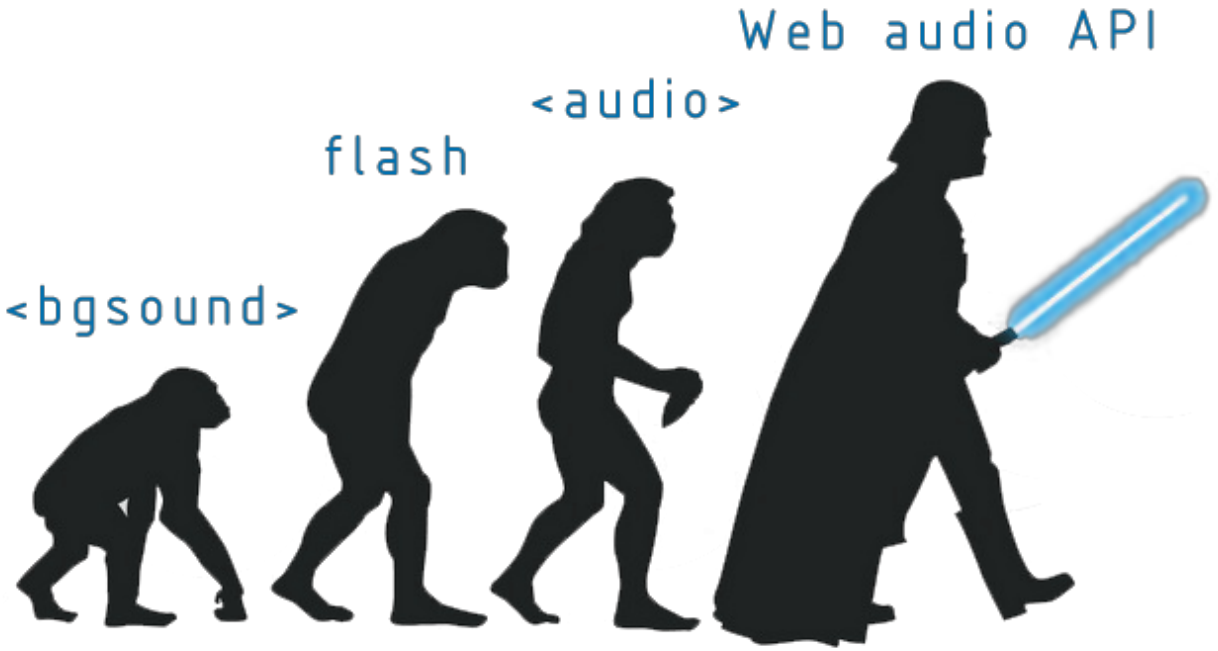
Web audio API

WSD Москва 2013

Дмитрий Дудин

@html5by

Эволюция звука в WEB



WebAudioAPI не <audio>

Web audio API и **<audio>**

почти никак не связаны

применение

<audio>

- Аудио плееры
- Фоновая музыка
- Аудио подсказки

применение

WEB AUDIO API

- 3D звук (игры)
- Обработка звука
- Синтез звука
- Визуализация
- Анализ звука

. . .

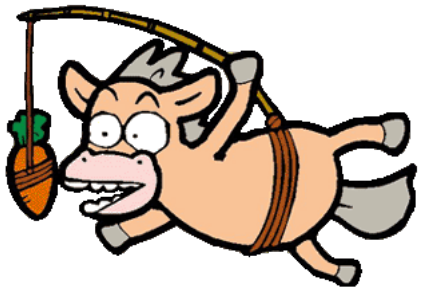
- Многоканальное аудио.
  API поддерживает
  до 32 каналов

- Точная синхронизация
  звуков во времени

- Интеграция с webRTC, webGL

А что с этой штукой можно делать?

goo.gl/MSCvq - online band
goo.gl/NnTNXS - 3D игры
goo.gl/sM4me - анализ
goo.gl/9ihlnh - обработка
moveit.html5.by - визуализация



Проблемы

Спецификация меняется

new **webkitAudioContext** -> new **AudioContext**()

AudioContext.**createDelayNode()** - > AudioContext.**createDelay()**

AudioBufferSourceNode.**noteOn(0)** -> AudioBufferSourceNode.**start()**

Решение:

https://github.com/cwilso/AudioContext-MonkeyPatch

Поддержка браузерами
декабрь 2013

отлично

Chrome, Safari, Opera,
Chrome android, Safari iOS

почти

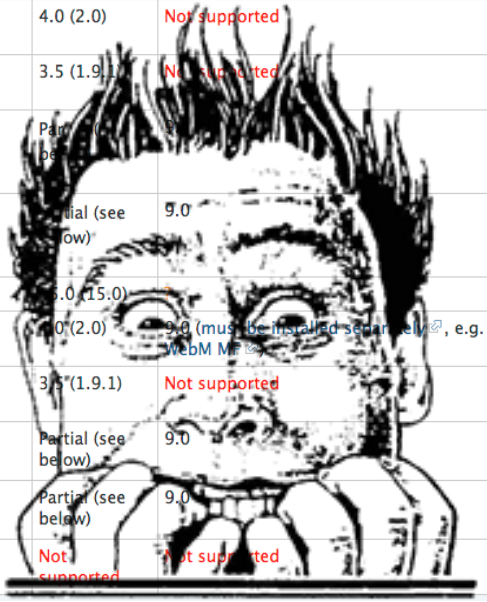
Firefox (desktop + mobile)

думают о поддержке в будущем

Internet explorer, blackberry и тп

Пока нет универсального аудио формата

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	3.0	3.5 (1.9.1)	9.0	10.50	3.1
<audio>: WAVE, PCM	(Yes)	3.5 (1.9.1)	Not supported	Not supported	3.1
<audio>: WebM, Vorbis	(Yes)	4.0 (2.0)	Not supported	10.60	3.1 (must be installed separately)
<audio>: Ogg, Vorbis	(Yes)	3.5 (1.9.1)	Not supported	10.50	3.1 (must be installed separately, e.g. XiphQT)
<audio>: MP4, MP3	(Yes) (Not in Chromium)	Partial (see below)	9.0	Not supported	3.1
<audio>: MP4, AAC	(Yes) (Main only) (Not in Chromium)	Partial (see below)	9.0	Not supported	3.1
<audio>: Ogg, Opus	27.0	15.0 (15.0)	?	?	?
<video>: WebM, VP8, Vorbis	(Yes)	4.0 (2.0)	9.0 (must be installed separately, e.g. WebM MF)	10.60	3.1 (must be installed separately, e.g. Perian)
<video>: Ogg, Theora, Vorbis	(Yes)	3.5 (1.9.1)	Not supported	10.50	3.1 (must be installed separately, e.g. XiphQT)
<video>: MP4, H.264, MP3	(Yes) (Not in Chromium)	Partial (see below)	9.0	Not supported	3.1
<video>: MP4, H.264, AAC	(Yes) (Not in Chromium)	Partial (see below)	9.0	Not supported	3.1
any other format	Not supported	Not supported	Not supported	Not supported	3.1 (plays all formats available via QuickTime)



sources
Источники звука для Web audio

- **AudioBufferSourceNode**
  - буффер

- **MediaElementAudioSourceNode**
  - <audio>,<video>

- **MediaStreamAudioSourceNode**
  - микрофон
  - любой stream

destinations

Получатели звука в Web audio

- **context.destination**
  - системные динамики

- **MediaStreamAudioDestinationNode**
  - stream (аналогичный getUserMedia)

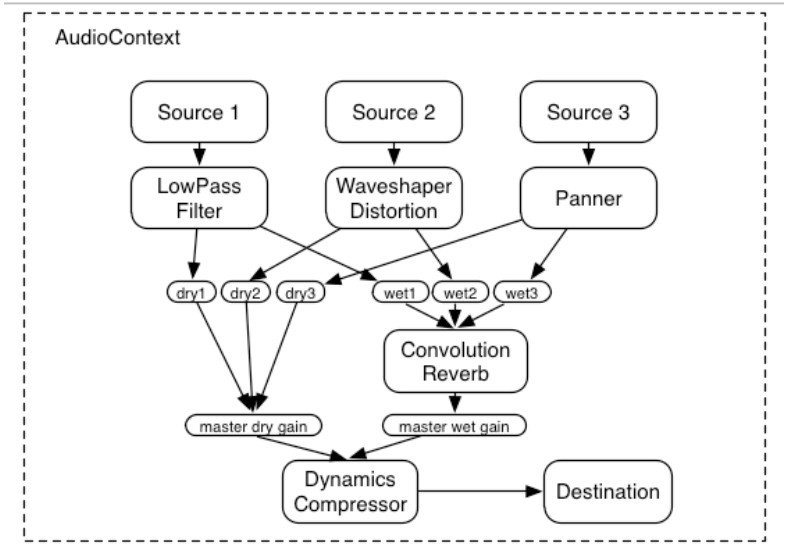
Audio context

один на всю страницу

и этого достаточно

1. множество
   входов
   и выходов аудио

2. сложные аудио
   графы



Все что нужно для начала работы

1. Создаем аудиоконтекст

```
var context = new AudioContext();
```

2. Загружаем звук в буфер

(например с XMLHttpRequest)

```
var loadSoundFile = function(url) {
  var xhr = new XMLHttpRequest();
  xhr.open('GET', url, true);
  xhr.responseType = 'arraybuffer';
  xhr.onload = function(e) {
    context.decodeAudioData(this.response,
    function(decodedArrayBuffer) {
      buffer = decodedArrayBuffer;
    }
  };
  xhr.send();
}
```

3. Соединяем и готово !

```
var play = function(){
  source = context.createBufferSource();
  destination = context.destination;
  source.buffer = buffer;
  source.connect(destination);
  source.start(0);
}

var stop = function(){
  source.stop(0);
}
```

Play

Строим схемы обработки звука

node.connect(otherNode)

```
source1.connect(node1);
source2.connect(node3);
node1.connect(node4);
node1.connect(node2);
node2.connect(destination);
node3.connect(node1);
node4.connect(destination);
node4.connect(node3);
```

Web audio API
содержит огромное количество
готовых к использованию
высокоуровневых
конфигурируемых
блоков

... как LEGO

нужно только соединить все правильно

Начнем с простого
усилитель (gain)
задержка (delay)

… и немного кода

Gain node

```
var gainNode = context.createGain();
gainNode.gain.value = 0.4; // 0..1

source.connect(gainNode);
gainNode.connect(destination);
```

Gain level

Play

Crossfade effect

2 tracks with gain nodes

```
gainNodes = [context.createGain(), context.createGain()];
gainNodes[0].gain.value  = 1 - $('.range1').val();
gainNodes[1].gain.value  = $('.range1').val();
```

Gain level

Play

Delay node

```
var delayNode = context.createDelayNode(10); // max possible delay
delayNode.delayTime.value = $('.range1').val();

source.connect(delayNode);
delayNode.connect(destination);
```

Delay time

Play

Echo effect with delay and gain

```
source.connect(gainNode);
gainNode.connect(destination);
gainNode.connect(delayNode);
delayNode.connect(gainNode);

now = context.currentTime;
source.start(now);
source.stop(now + 0.3);
```

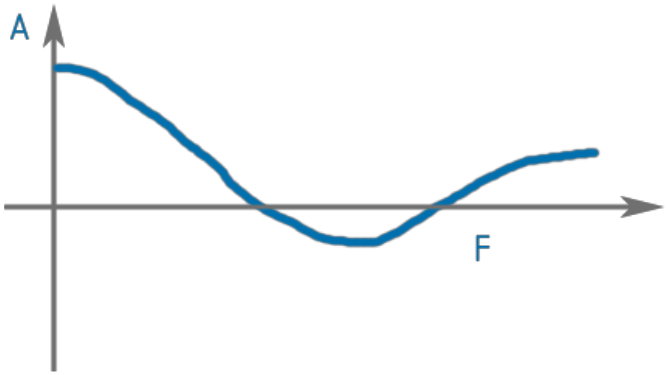
Play

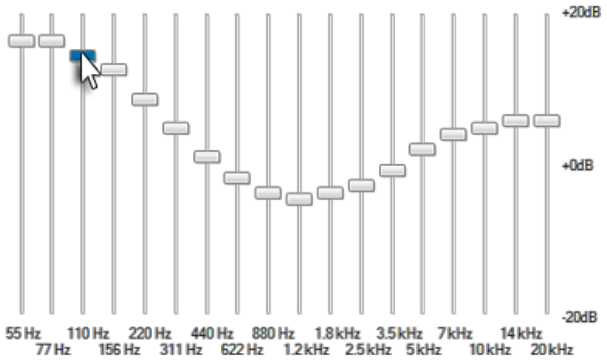
Фильтры

Амплитудно-частотная характеристика
(Frequency response function)

≈

Установки эквалайзера любимом плеере
(бас, диско, вокал...)





.... эквалайзер да и только

Biquad filter node

```
var filterNode = context.createBiquadFilter();
filterNode.type = 1;                // High-pass filter.
filterNode.frequency.value = 1000; // Cutoff to 1kHZ
filterNode.frequency.Q = 1;         // QUality factor
filterNode.gain.value = 1000;       // Not used in this filter type
```

◉ Lowpass filter ○ Highpass filter ○ Bandpass filter

○ Lowshelf filter ○ Highshelf filter ○ Peaking filter ○ Notch filter ○ All-pass filter

Frequency

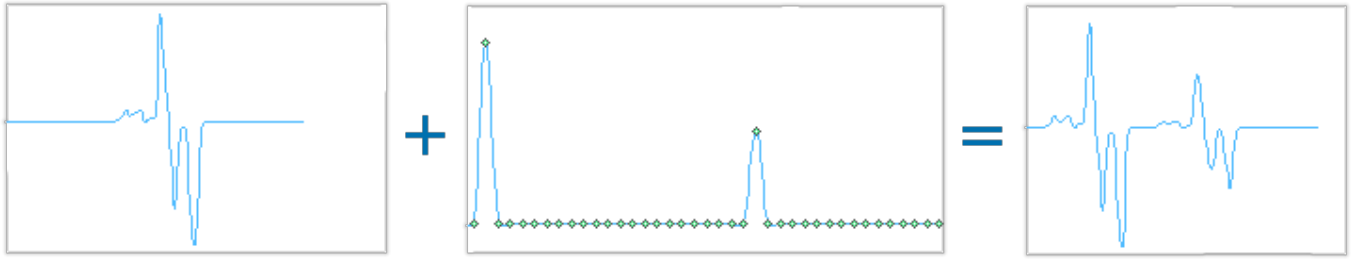
Quality factor

Gain

Play

Свертка
Convolution

пример: эхо



Входной сигнал

Импульсная характеристика (impulse response)

Выходной сигнал

Convolver node

```
convolverNode = context.createConvolver();
convolverNode.buffer = buffer; // impulse response

source.connect(convolverNode);
convolverNode.connect();
```

◉ Wheel ○ 1950 ○ Basement ○ Telephone ○ Muffer ○ Spring ○ Echo

Play

Генератор
(oscilator)

Oscilator node (with analyser)

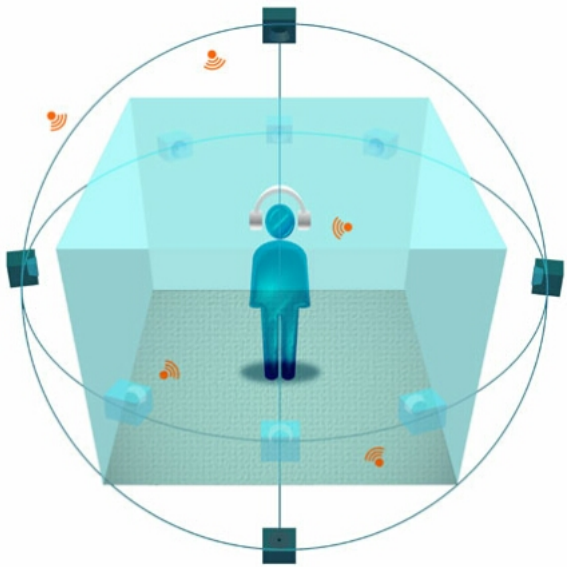
```
oscillator = context.createOscillator();
analyser = context.createAnalyser();

oscillator.connect(analyser);
analyser.connect(destination);

oscillator.start(0);
```

◉ Sine ○ Square ○ Sawtooth ○ Triangle

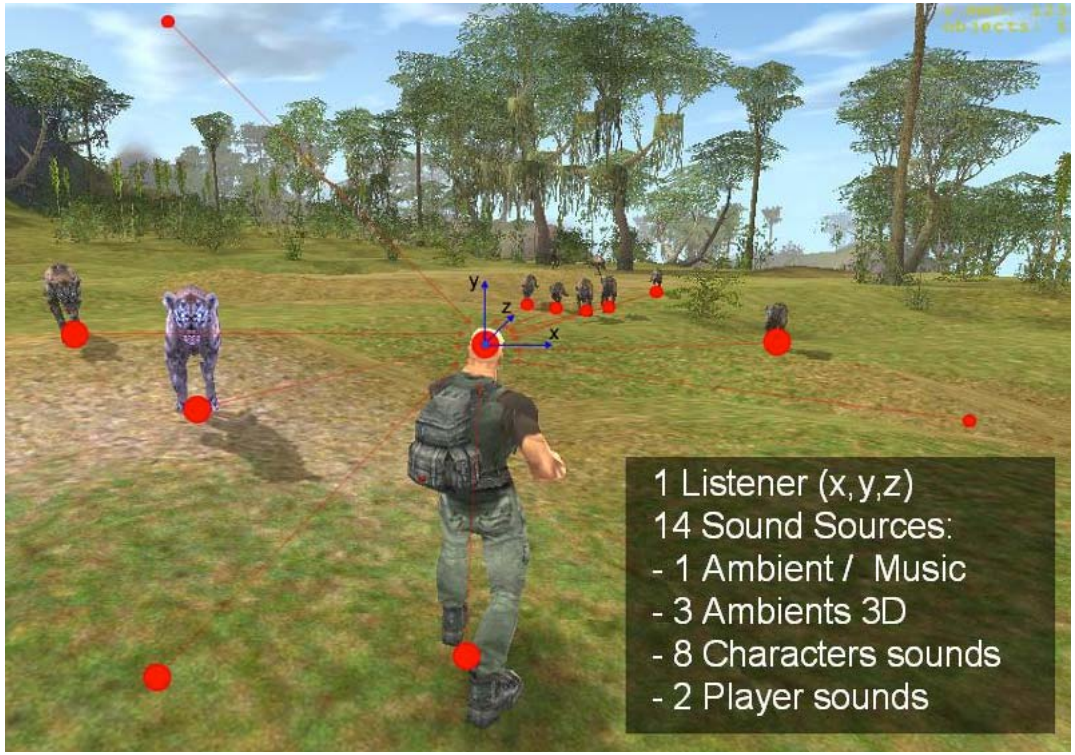
Frequency Detune

Play

3D



Типовая ситуация



Panner

У каждого паннера есть

- координаты
- направление
- скорость

У конечного слушателя есть

- координаты
- направления
- скорость

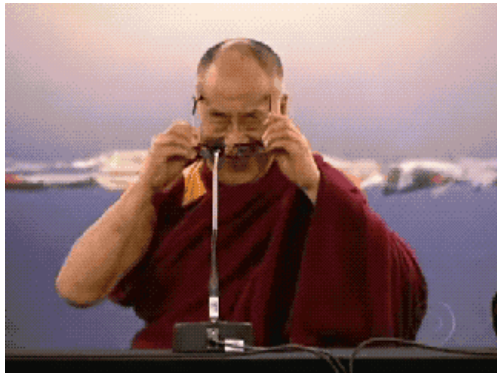
goo.gl/5afn3n

И многое другое

- Analyser - все данные о времени и частоте
- ChannelSplitterNode - разделение каналов
- ChannelMergerNode - объединение каналов
- DynamicsCompressorNode - динамический компрессор
- WaveShaperNode - нелинейные искажения
- ScriptProcessorNode - можно делать все что хотим

Демки

- goo.gl/szY7Ko - web audio toy
- goo.gl/LSJlXv - web audio playground
- goo.gl/XuH0bh - midi drum mashine
- goo.gl/UVbLMt - piano
- goo.gl/5gm5PR - rally
- goo.gl/YdFlWc - xylophone
- goo.gl/qHY9p7 - и еще много разных демок



@html5by

html5.by/blog/audio

Вопросы?