





Making music mobile

#webrtc

Photo source: [Feliciano Guimarães](#)



Justin Uberti
+Justin Uberti
@juberti



Per Emanuelsson
+Soundtrap
@soundtrapsite





Photo source: [Jonathan Athayde](#), image cropped



Photo source: [Rebecca Wilson](#), image cropped



Media Capture

Photo source: [Staffan Vilcans](#)



`getUserMedia()`



Audio
track

MediaStream



`getUserMedia()`



AEC

AGC

Audio
track

MediaStream

AEC: Acoustic Echo Cancellation

AGC: Auto Gain Control



getUserMedia()



AEC

AGC

Audio track

MediaStream

```
navigator.getUserMedia({audio: true}).then((stream) {  
  // do something  
});
```




getUserMedia()



Audio track

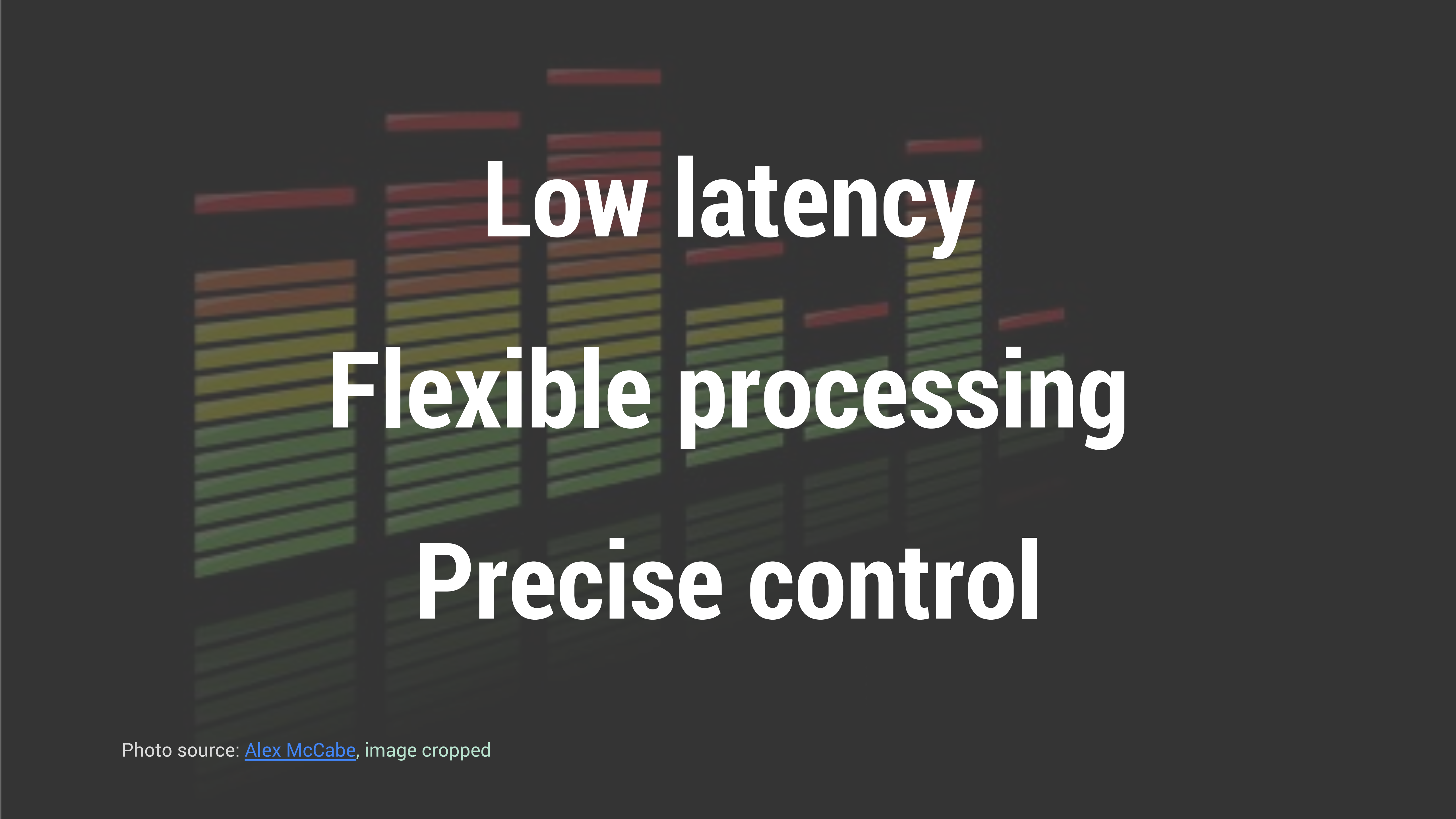
MediaStream

```
navigator.getUserMedia({audio: {  
  echoCancellation: false  
}}).then((stream) {  
  // do something  
});
```




Web Audio

Photo source: [Alex McCabe](#), image cropped



Low latency
Flexible processing
Precise control

Photo source: [Alex McCabe](#), image cropped

Dynamics processing

3D positioning

Acoustic environments

Event scheduling

Oscillators

Fades & sweeps

Filtering effects

Waveform analysis

Low latency

Frequency analysis

WebRTC Integration

Waveshaping

Doppler shift

The Web Audio pipeline

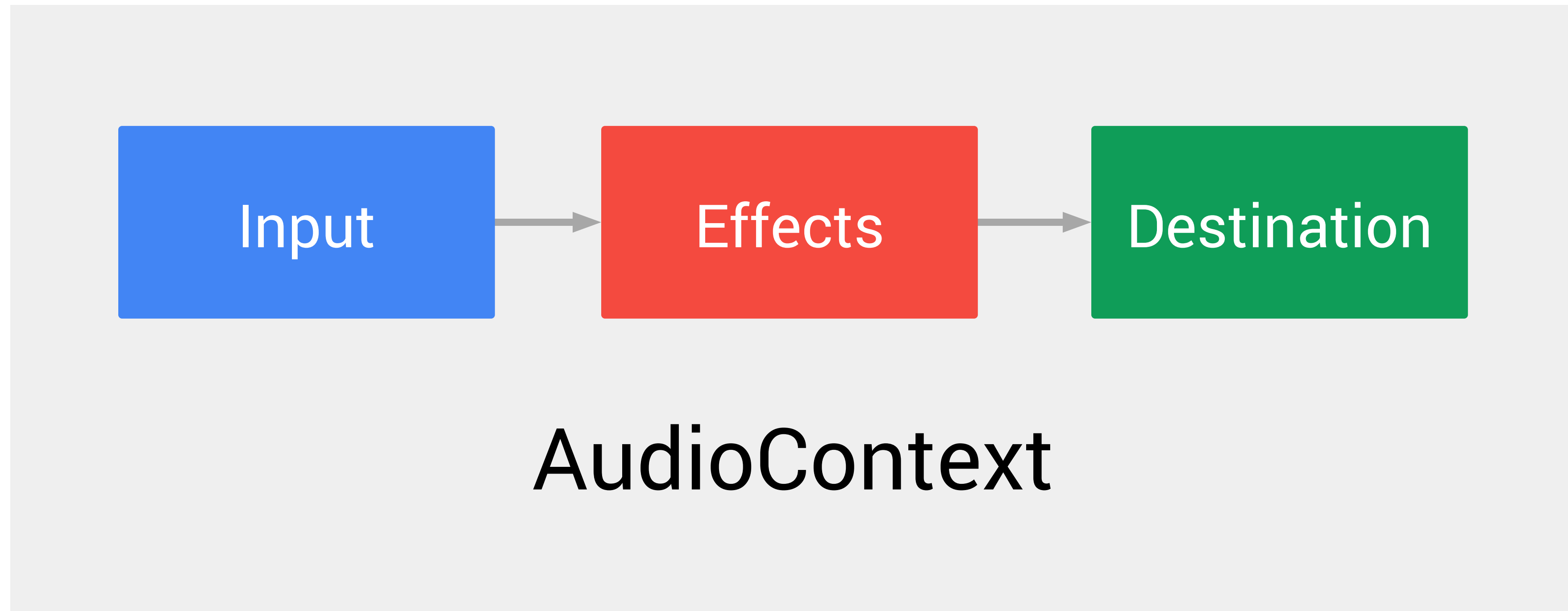
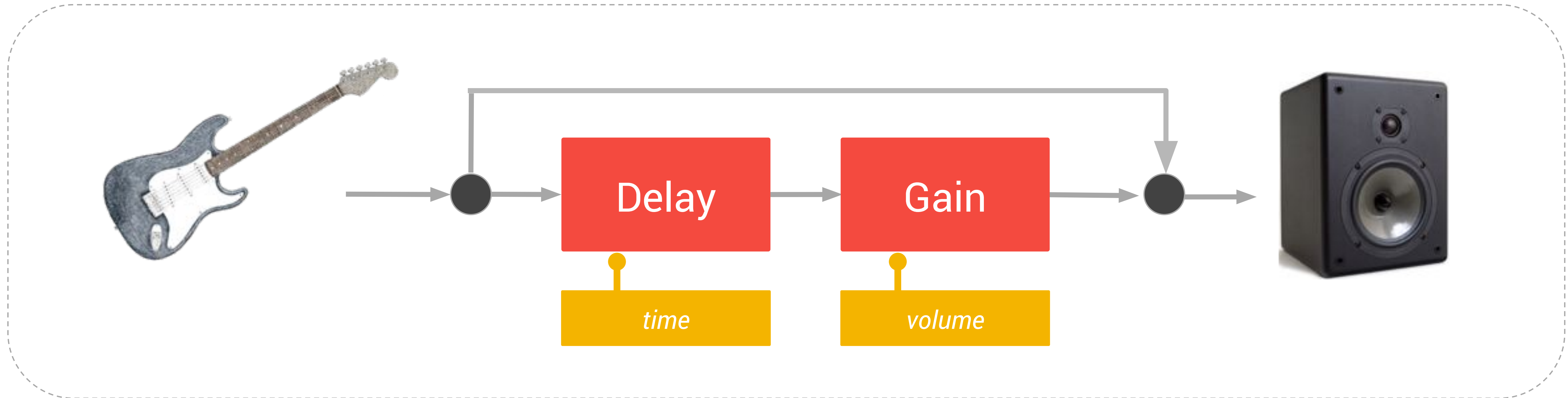




Photo source: [Michael Morel](#), image cropped

Slapback effect (Elvis)



Adds a delayed copy with lower volume



```
// direct audio path
```

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

```
navigator.getUserMedia(audio:true)
```



```
// direct audio path
```

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

```
navigator.getUserMedia(audio:true)
```

```
.then((stream) {
```

```
    var guitar = context.createMediaStreamSource(stream);
```

```
    var speaker = context.destination;
```

```
});
```




```
// direct audio path
```

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

```
navigator.getUserMedia(audio:true)
```

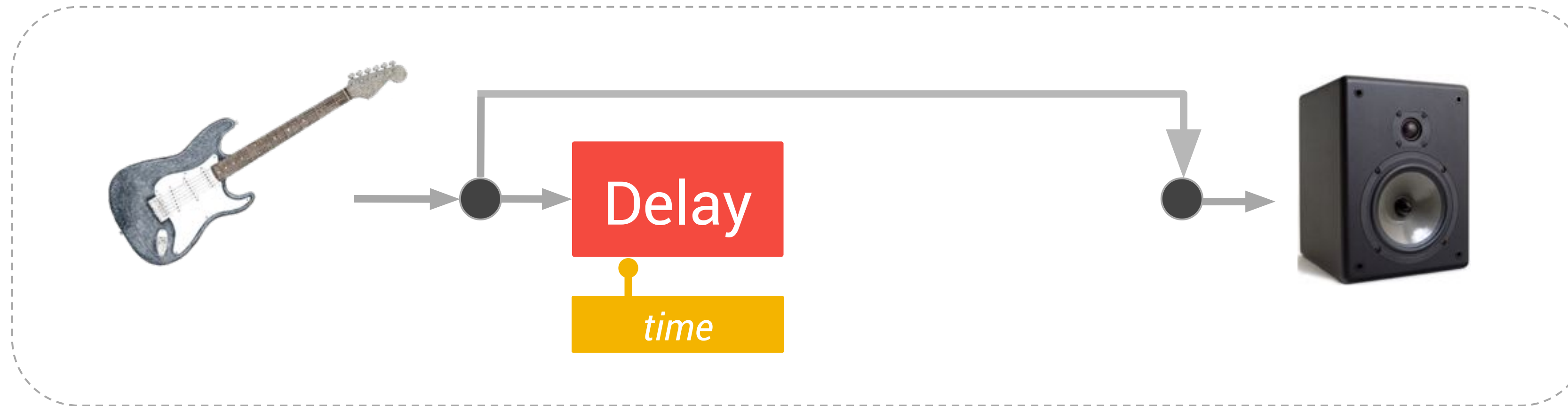
```
.then((stream) {
```

```
    var guitar = context.createMediaStreamSource(stream);
```

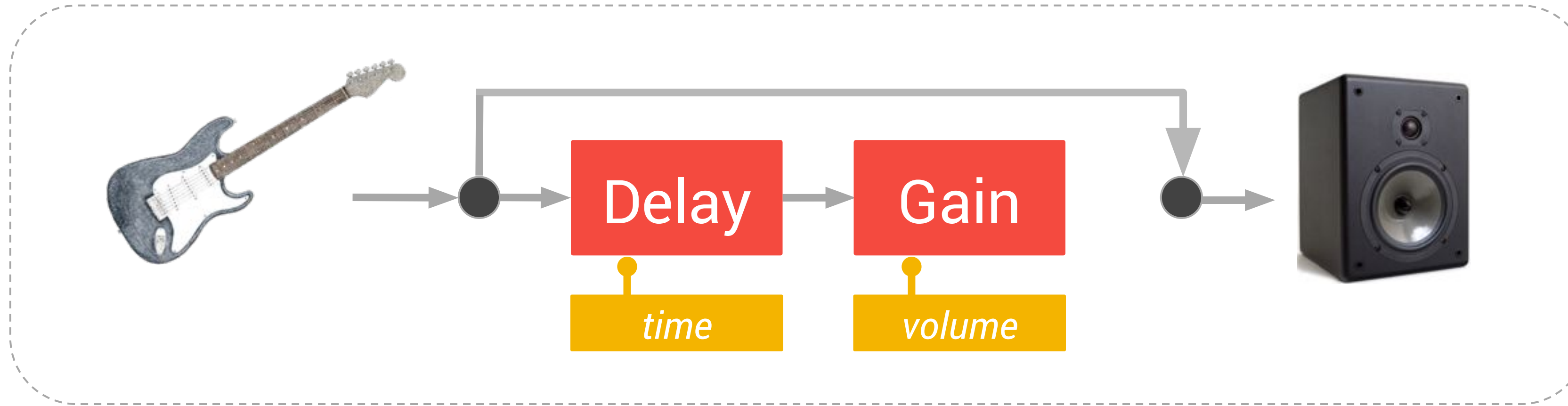
```
    var speaker = context.destination;
```

```
    guitar.connect(speaker);
```

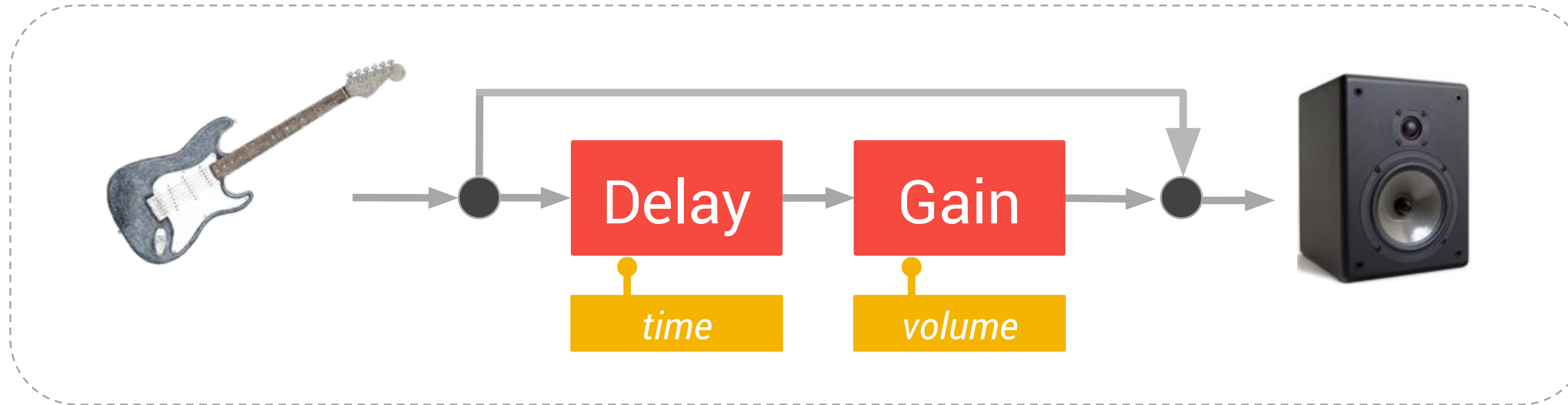
```
});
```



```
// delayed audio path  
var delay = context.createDelay();  
guitar.connect(delay);
```

```
// delayed audio path  
var delay = context.createDelay();  
guitar.connect(delay);  
  
var gain = context.createGain();  
delay.connect(gain);
```



```
// delayed audio path
```

```
var delay = context.createDelay();  
guitar.connect(delay);
```

```
var gain = context.createGain();  
delay.connect(gain);
```

```
gain.connect(speaker);
```



```
// direct audio path  
var context = new AudioContext();  
navigator.getUserMedia(audio:true)  
  .then((stream) {  
    var guitar = context.createMediaStreamSource(stream);  
    var speaker = context.destination;  
    guitar.connect(speaker);  
  });
```

```
// delayed audio path  
var delay = context.createDelay();  
guitar.connect(delay);
```

```
var gain = context.createGain();  
delay.connect(gain);
```

```
gain.connect(speaker);
```

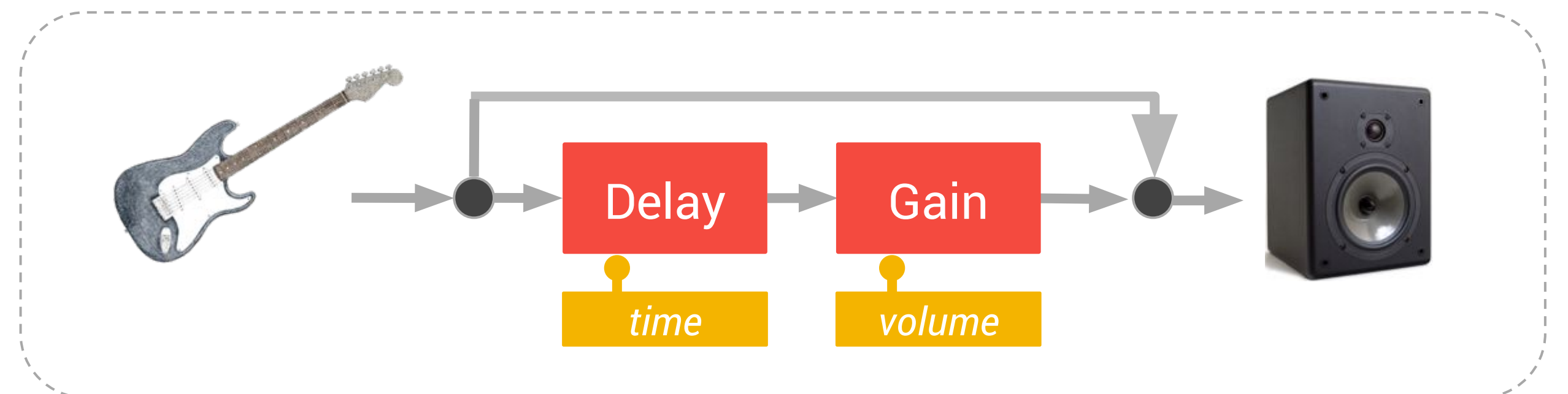




Photo source: [Blondin Rikard](#), image cropped

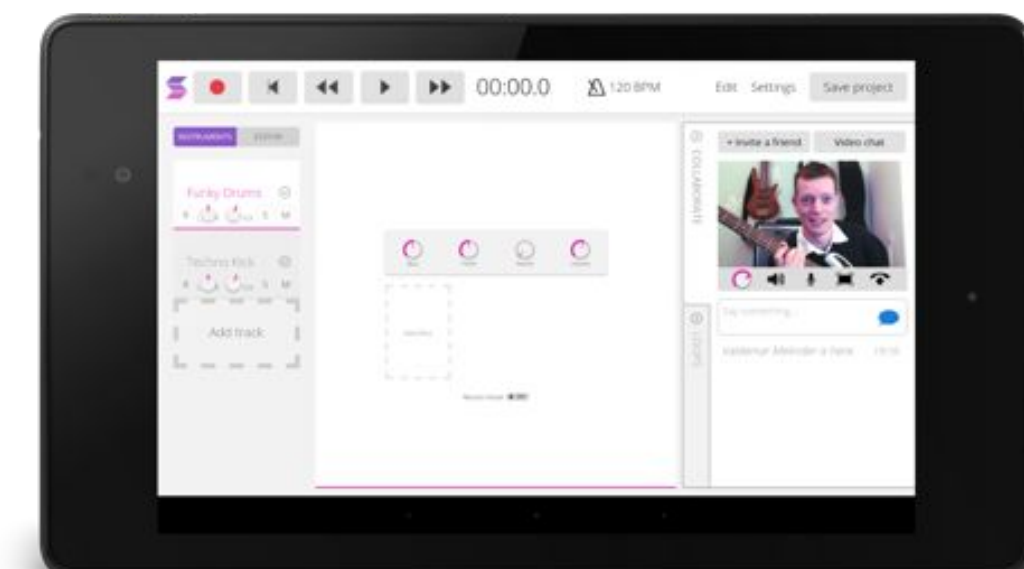


WebRTC



Realtime peer to peer audio, video, data

Photo source: [Eric Fischer](#)



Peer
connection

Audio
+
Video



Audio
+
Video

Peer
connection

MediaStream
↑
getUserMedia()

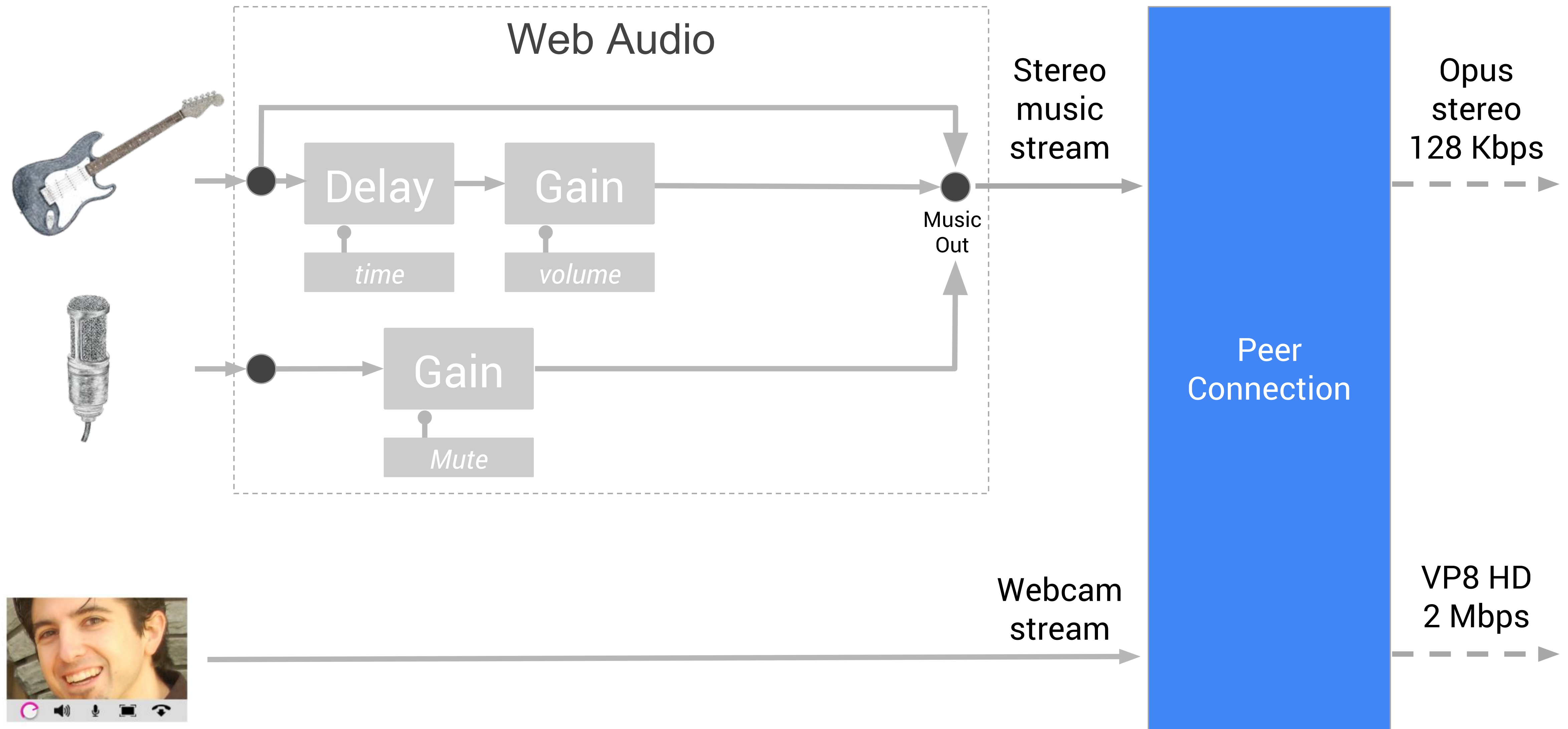
MediaStream
↑
getUserMedia()

The background image is a screenshot of the Soundtrap web application interface. It shows a project titled 'Funkyzeit-freakout-bonanza!' in 'PRO MODE'. The interface includes a left sidebar with instrument tracks like Bass, Rhythm guitar, Keyboard, Drums, and Acoustic guitar, each with a MIDI piano roll. The main workspace has a timeline with several audio and MIDI clips. At the top, there are navigation buttons for 'Edit', 'Settings', and 'Save project'. At the bottom, there is a playback control bar with buttons for record, play, stop, and a tempo display of 110 BPM.

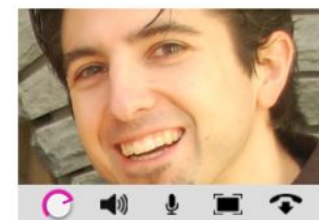
Video chat

High-quality stereo audio

Input from Web Audio



```
navigator.getUserMedia(video:true).then((webcamStream) {  
  
    // get music stream  
  
    // create peer connection  
  
    // start call  
  
});
```



Webcam
stream




```
navigator.getUserMedia(video:true).then((webcamStream) {
```

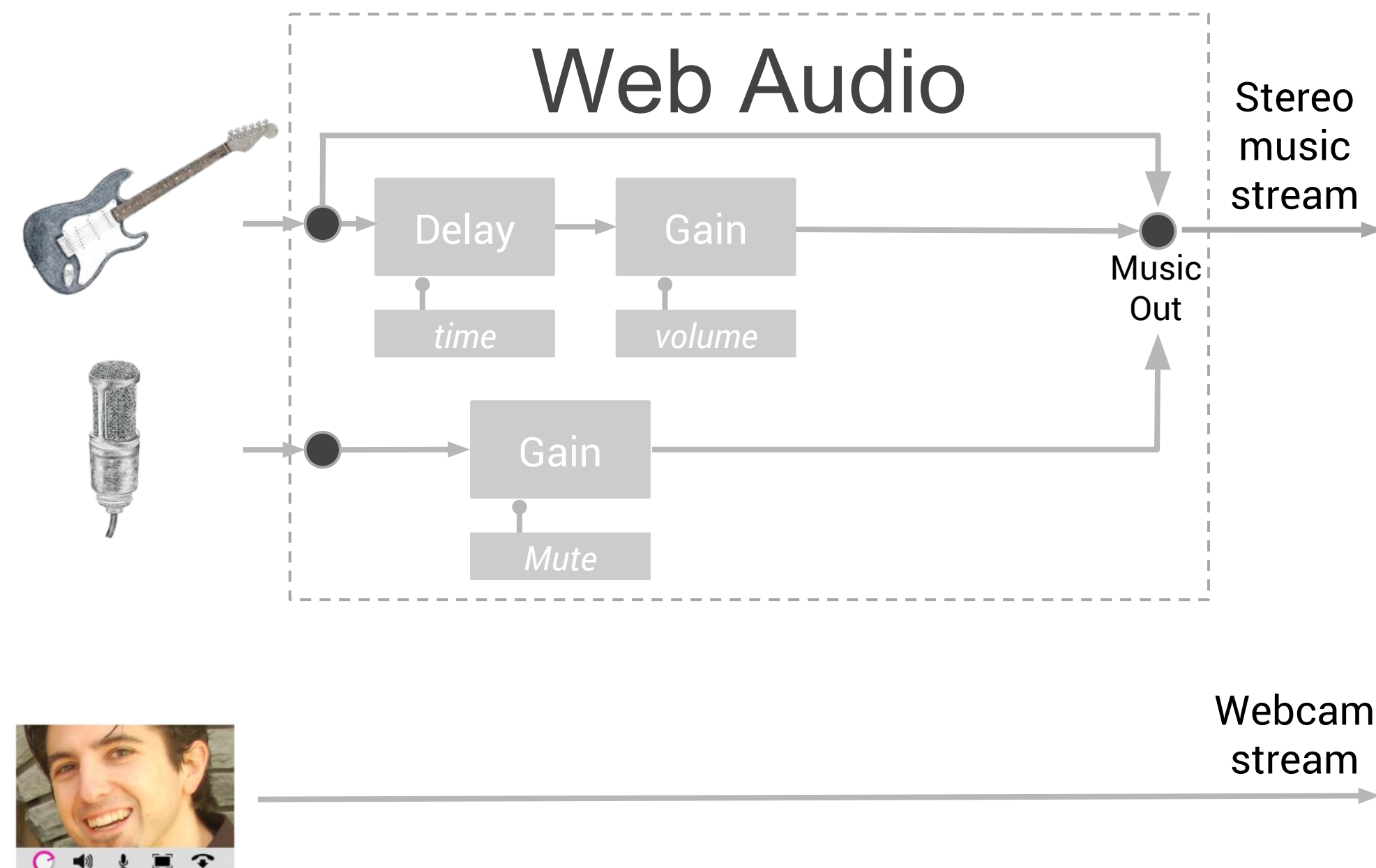
```
    var dest = context.createMediaStreamDestination(musicOut);
```

```
    var stereoMusicStream = dest.stream;
```

```
// create peer connection
```

```
// start call
```

```
});
```



```
navigator.getUserMedia(video:true).then((webcamStream) {
```

```
var dest = context.createMediaStreamDestination(musicOut);
```

```
var stereoMusicStream = dest.stream;
```

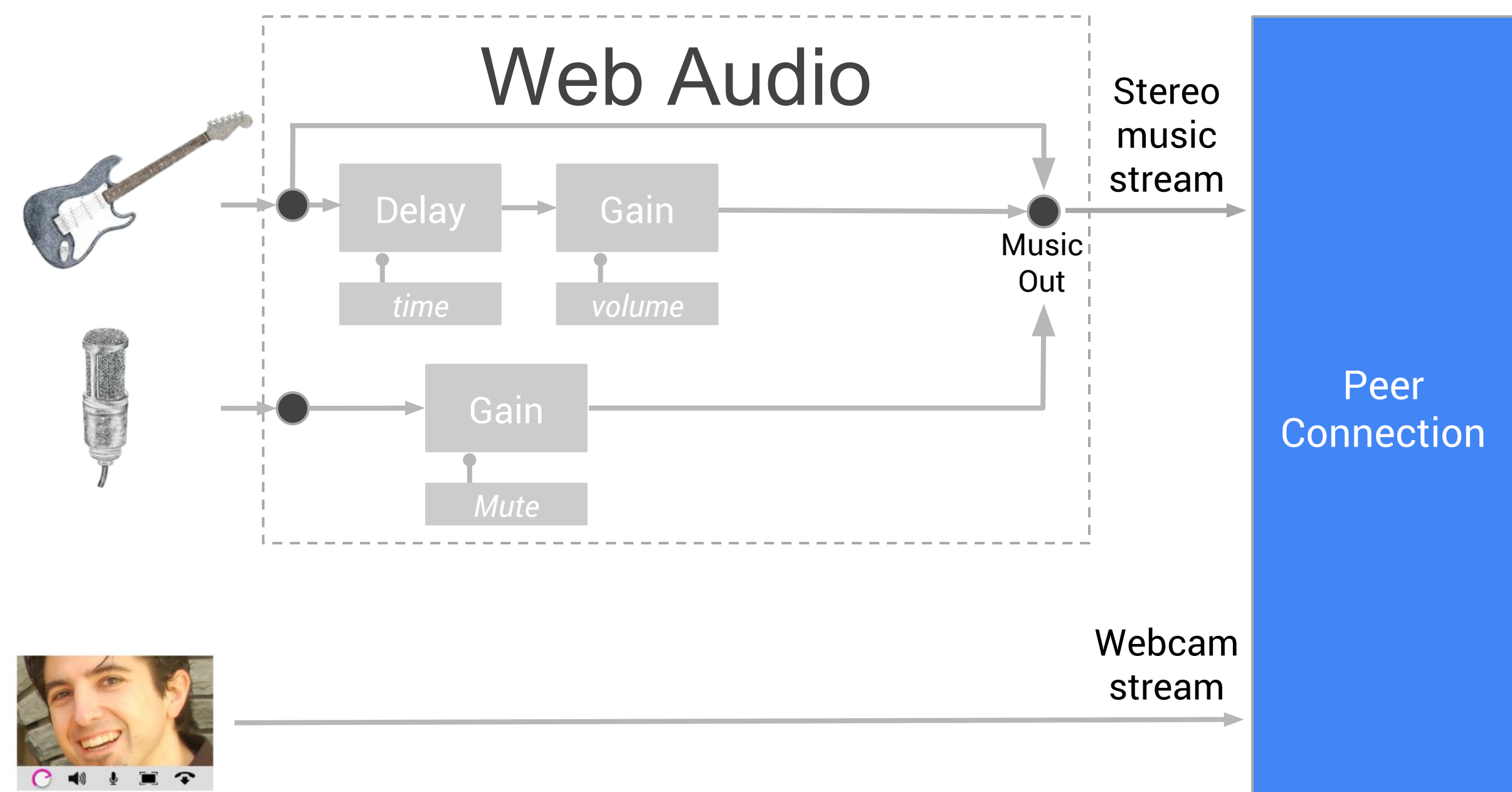
```
var peerConnection = new RTCPeerConnection()
```

```
..addStream(stereoMusicStream)
```

```
..addStream(webcamStream);
```

```
// start call
```

```
});
```




```
navigator.getUserMedia(video:true).then((webcamStream) {
```

```
var dest = context.createMediaStreamDestination(musicOut);
```

```
var stereoMusicStream = dest.stream;
```

```
var peerConnection = new RTCPeerConnection()
```

```
..addStream(stereoMusicStream)
```

```
..addStream(webcamStream);
```

```
startCall(peerConnection);
```

```
});
```

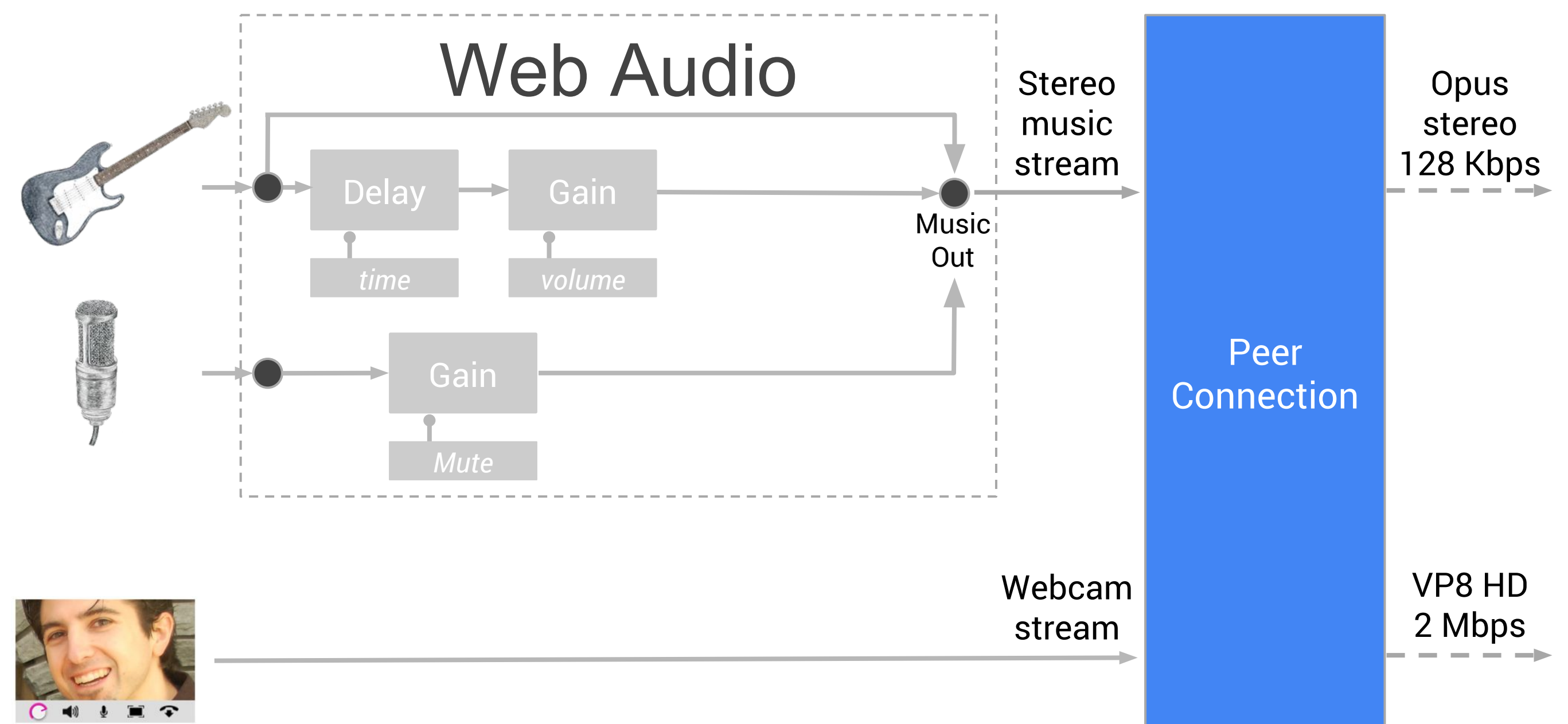


Photo source: [Ulisse Albiati](#)



Familiarity

Scalability

Productivity



Dart




```
import 'dart:async' show Future;
```

```
class User {
```

```
  String username;
```

```
  String password;
```

```
  User(this.username, this.password);
```

```
  User.fromJson(Map data) :
```

```
    username = data['username'],
```

```
    password = data['password'];
```

```
  bool get isValidUsername => username != null;
```

```
  static Future<User> load(Service backend, int id) {
```

```
    //
```

```
  }
```

```
}
```

```
import 'dart:async' show Future;
```

```
class User {
```

```
    String username;
```

```
    String password;
```

```
    User(this.username, this.password);
```

```
    User.fromJson(Map data) :
```

```
        username = data['username'],
```

```
        password = data['password'];
```

```
    bool get isValidUsername => username != null;
```

```
    static Future<User> load(Service backend, int id) {
```

```
        //
```

```
    }
```

```
}
```



```
import 'dart:async' show Future;
```

```
class User {
```

```
    String username;
```

```
    String password;
```

```
    User(this.username, this.password);
```

```
    User.fromJson(Map data) :
```

```
        username = data['username'],
```

```
        password = data['password'];
```

```
    bool get isValidUsername => username != null;
```

```
    static Future<User> load(Service backend, int id) {
```

```
        //
```

```
    }
```

```
}
```

Functions

Classes

Libraries

Packages




```
import 'dart:async' show Future;

class User {
  String username;
  String password;

  User(this.username, this.password);
  User.fromJson(Map data) :
    username = data['username'],
    password = data['password'];

  bool get isValidUsername => username != null;

  static Future<User> load(Service backend, int id) {
    // do something later
  }
}
```

```
20
21=main() {
22     var user = new User('Bob', 'verifyme');
23     print(user.username);
24 }
```

Problems

⚠ There is no such getter 'username' in 'User'

Static type

dynamic

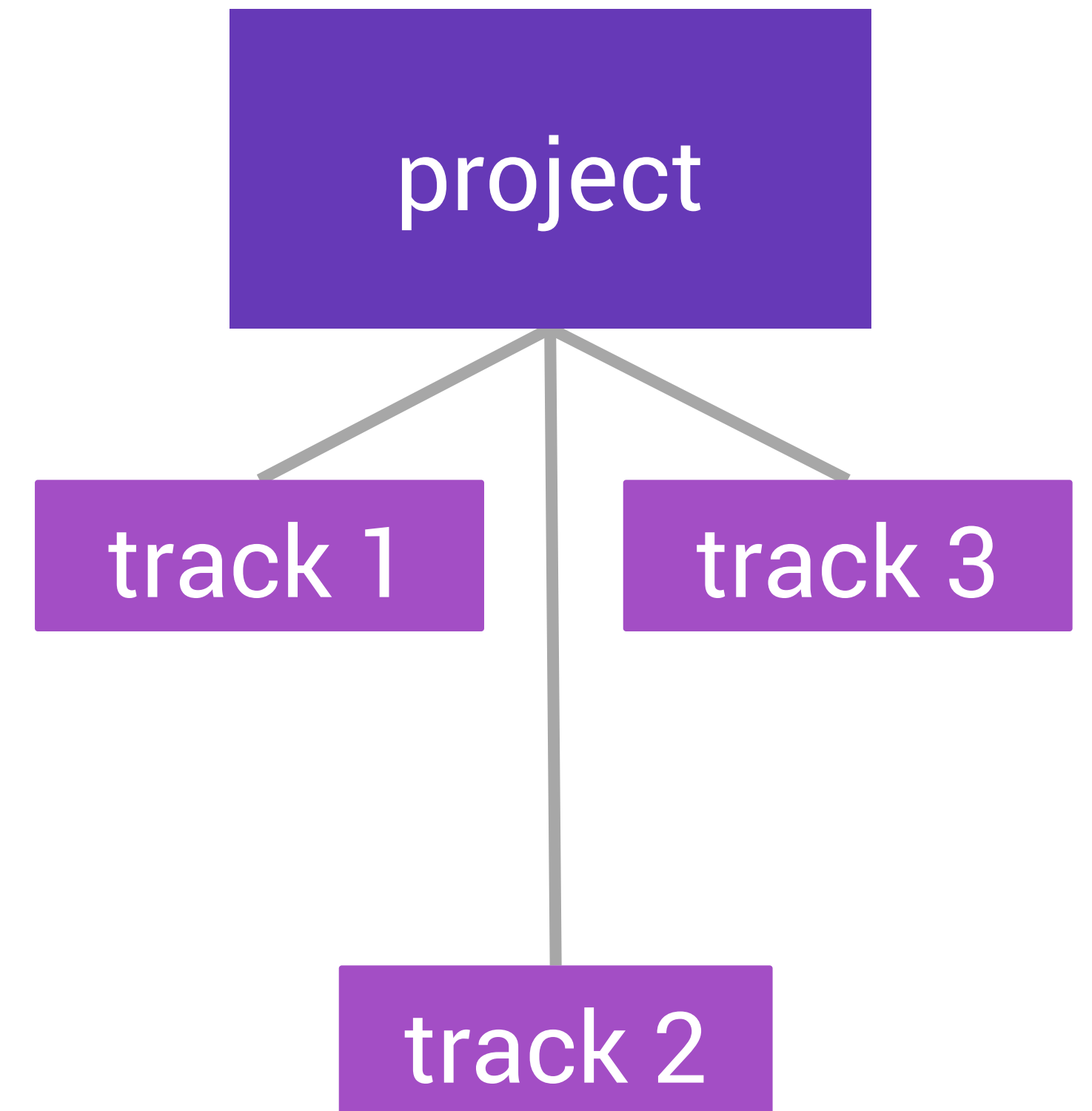
Parameter

Object object

Loading a project

1. Load project asynchronously using Futures:

```
Future loadProject() => httpRequest.request(...)  
                        .then( /* parse json */ )
```



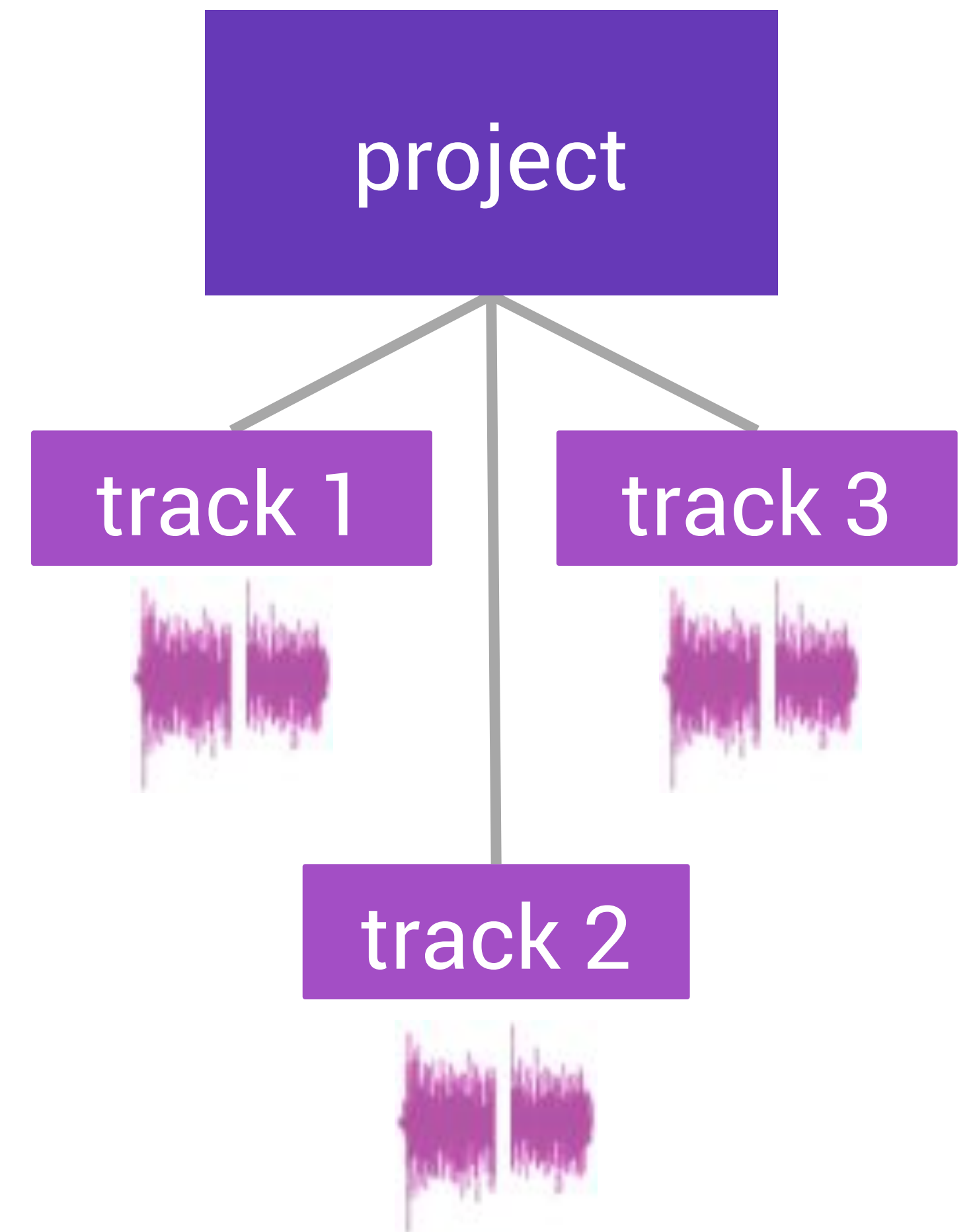
Loading a project

1. Load project asynchronously using Futures:

```
Future loadProject() => httpRequest.request(...)  
                        .then( /* parse json */ )
```

2. For every track, get audio data:

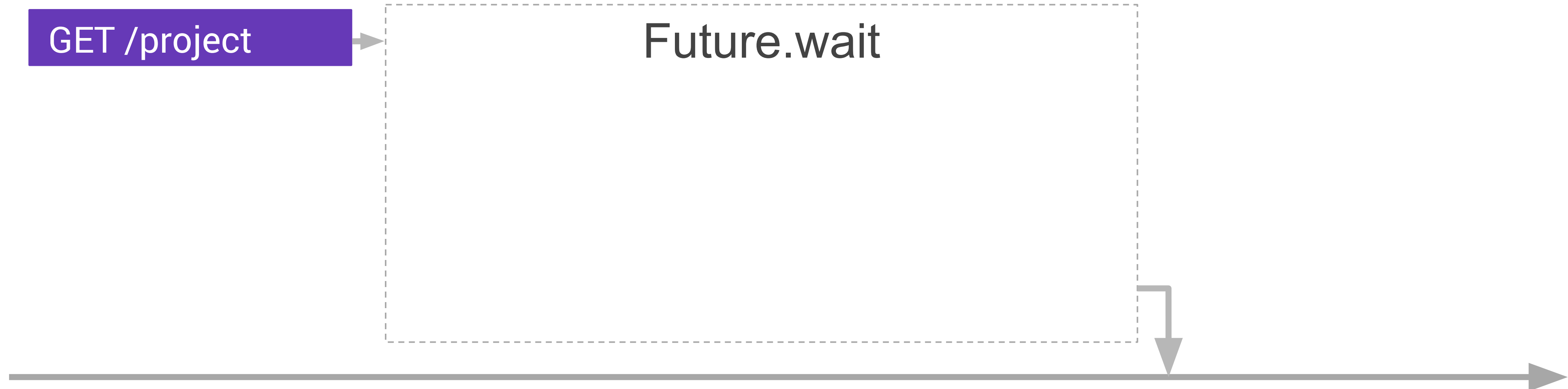
```
Future loadAudio(Track t) => httpRequest.request(...)  
                        .then( /* decode audio */ )
```



Load all audio tracks simultaneously

```
loadProject()
```

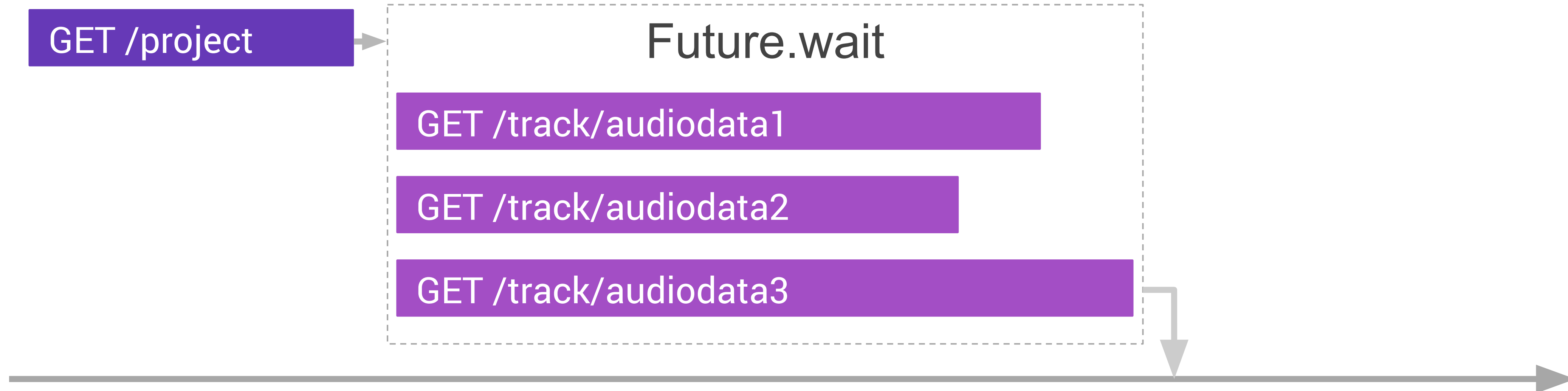
```
.then((project) => Future.wait( /* list of Futures */ ))
```



Load all audio tracks simultaneously

```
loadProject()
```

```
.then((project) => Future.wait(project.tracks.map(loadAudio)))
```



Built with the modern Web

Audio mixer

Effects

Synth

Sampler

Video chat

Instrument
controllers

Recorder

Piano roll
editor

Offline
renderer

Load/save

Web Audio

Web MIDI

WebRTC

WebSocket

<video>

Web Audio

2B devices

1 B mobile



in development



WebRTC

1.5B devices

300M mobile



under consideration



Android WebView

Web Audio + WebRTC





soundtrap.com/io2014

Visualizer by [Felix Turner](#)

Further listening

Getting Started with WebRTC

html5rocks.com/en/tutorials/webrtc/basics

Getting Started with the Web Audio API

html5rocks.com/en/tutorials/webaudio/intro

Get Started with Dart

www.dartlang.org/docs/tutorials/get-started/

youtube.com/GoogleDevelopers



We want to hear from you!

Session feedback

<http://goo.gl/QOKHZI>



Soundtrap

soundtrap.com

