DM2: Manipulation de flux audio avec l'API Web Audio

Introduction

Nous avons vu en cours qu'un document HTML5 permet de distribuer un média audio ainsi que de le manipuler via le DOM (vitesse de lecture, volume, ...). Cependant, cette API ne permet pas d'effectuer des manipulations du flux audio...

L'API Web Audio, recommandation candidate du W3C, propose un ensemble d'objets de haut-niveau pour contrôler en temps réel des données audio. Elle permet à partir de sources audio de les analyser, les visualiser, d'ajouter des filtres d'effets ou encore d'appliquer des effets de spatialisation. Elle est implémentée dans les version récentes de Firefox, Chrome, Safari ainsi qu'Opera.

Ce projet a pour but de vous familiariser avec l'apprentissage et utilisation d'une API Web disponible en JavaScript. Vous devrez concevoir une page dynamique permettant pour une ressource audio donnée de la jouer, d'obtenir des informations ainsi que d'appliquer des effets à la demande.

Rappels de cours

HTML: Élément <audio>

```
<audio id="player" src="sample.wav" controls></audio>
```

Attributs

Les attributs HTML suivants sont disponibles pour l'élément <audio> :

- src: adresse de la ressource multimédia (URL)
- autoplay: commence la lecture dès que possible
- loop: lecture en boucle (booléen)
- controls: affichage des contrôles par le navigateur

JavaScript `DOM`: les objets `Audio`

Le chargement de la ressource média est **asynchrone** (comme les images).

```
// Création d'un nouvel élément...
var a = new Audio();
a.src = "sample.wav";
a.controls = true;
// ... ou récupération d'un élément existant
var a = document.getElementById("player");
```

Propriétés et méthodes

- État
  - `a.paused`/`a.ended` [R]: lecture en pause/finie (booléen)
  - `a.play()`/`a.pause()`: mise en lecture/pause
  - `a.playbackRate` [R/W]: vitesse de lecture (nombre: par défaut `1.0`)
- Position (nombre en secondes)
  - `a.duration` [R]: durée de lecture
  - `a.currentTime` [R/W]: position dans la lecture
- Contrôles
  - `a.volume` [R/W]: (nombre entre 0.0 et 1.0)
  - `a.muted` [R/W]: (booléen)
- Événements
  - `playing`, `ended`, `play`, `pause`, `waiting`, ...

Les tableaux typés en JavaScript

Les tableaux JavaScript (`Array`) peuvent être agrandis ou réduits dynamiquement et contenir des valeurs de types différents. Cependant, ce dynamisme implique un coût supplémentaire en performance et n'est pas adapté à certains cas d'utilisation (manipulation de flux audio et vidéo, WebSockets, ...).

Les tableaux typés, introduits dans la spécification JavaScript depuis `ECMAScript 2015`, permettent de répondre à ces cas d'utilisation. Ce sont des objets semblables à des tableaux (à la manière du langage C) qui permettent d'accéder de manière **typée** à des **tampons de mémoire** (*buffers*) de **taille fixe**.

Différents types de tableaux typés sont disponibles pour les types numériques usuels: `Uint8Array` pour des valeurs `Uint8` (entiers non signés sur 8 bits $\in [0, 255]$), `Float32Array` pour des valeurs `Float32` (flottants sur 32 bits), ...

- Création d'un tableau de `n` éléments de type `Uint8`:

```
let a = new Uint8Array(n) // soit n * 8 bits alloués
```

- Accès à un index du tableau:

```
a[i] = 32; // Attention: i doit être compris entre 0 et n-1
```

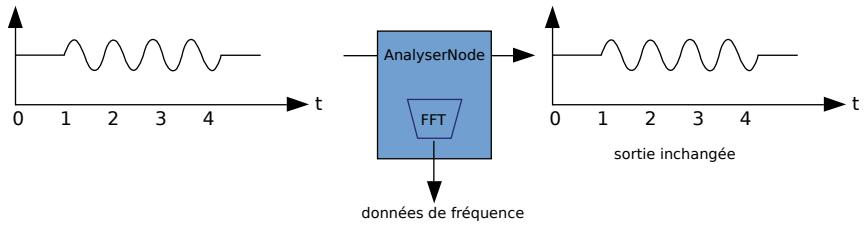
- Nombre d'éléments dans le tableau:

```
a.length; // vaut ici `n`
```

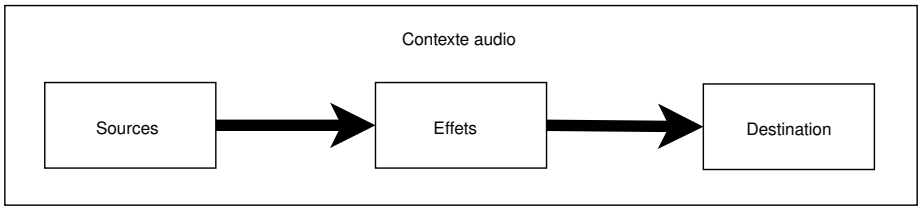
Web Audio API

La Web Audio API effectue des opérations dans un **contexte audio**. Dans ce contexte, les opérations audio (analyses, filtres, ...) sont réalisées par des **nœuds audio** reliés entre eux pour former un **graphe audio** orienté.

Chaque **nœud** est caractérisé par ses **entrées** de flux audio et ses **sorties** résultantes de l'**opération audio** effectuée. Par exemple, le nœud suivant a *une entrée* de flux audio ainsi qu'*une sortie* inchangée et permet de récupérer des données de fréquences:



Un nœud audio est un nœud **source** s'il décrit une source audio (par exemple l'élément `<audio>`). De manière similaire, un nœud **destination** représente la destination finale de l'audio du contexte (par exemple les hauts-parleurs de votre ordinateur).



Un processus de développement typique avec `Web Audio` ressemble à ceci:

- Création d'un contexte audio

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

- Dans ce contexte, création de la source (depuis l'élément `<audio>`, ...)

```
// 'a' est un élément 'Audio'
var src = ctx.createMediaElementSource(a);
```

- Création de nœuds de filtres d'effets

```
var analyser = ctx.createAnalyser();
// var node = ctx.createXXX(); // pour créer XXXNode
```

- Liaison des sources aux effets, et des effets à la destination

```
var dst = ctx.destination; // destination
src.connect(analyser); // source -> analyser
analyser.connect(dst); // analyser -> destination
```

`AudioContext`: contexte audio

Le contexte audio représente le **graphe audio** fait de nœuds audio reliés entre eux. Il gère l'exécution du traitement audio ainsi et permet de créer différents types de nœuds avec les méthodes `ctx.createXXX()` (où `XXXNode` est un type de nœud existant). Par exemple :

```
var analyser = ctx.createAnalyser(); // créé un AnalyserNode
...
```

Le nœud de **destination** est obtenu par la propriété

```
var dst = ctx.destination
```

`AudioNode`: propriétés communes aux nœuds audio

Chaque **nœud audio** peut être connecté et déconnecté à d'autres nœuds en utilisant les méthodes `connect(node)` et `disconnect(node)`:

```
src.connect(dst); // connecte le noeud src au noeud dst
src.disconnect(dst); // déconnecte le noeud src du noeud dst
```

`MediaElementAudioSourceNode`: source audio

Ce nœud audio source est obtenu depuis un élément média `<audio>` (et même `<video>` !). Il est obtenu par la méthode `ctx.createMediaElementSource(a)` où `a` est un objet `DOM` correspondant à un élément média.

- Nombre d'entrées: 0
- Nombre de sorties: 1

Attention: Certaines politiques de sécurité de navigateurs (`CORS`[1]) empêche d'utiliser un élément média qui a comme source une ressource sur votre ordinateur ou sur un autre serveur. Ajouter l'attribut `crossorigin=anonymous` à l'élément `<audio>` et fournir une ressource une ressource disponible depuis un serveur l'autorisant permet d'outrepasser cette limite.

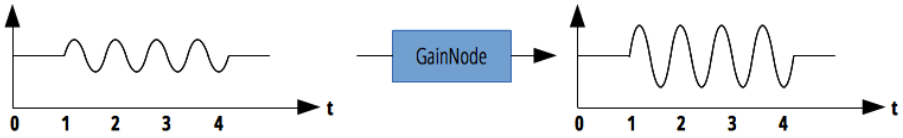
[1] https://miguelmota.com/blog/understanding-cross-origin-resource-sharing-cors/

`AnalyserNode`: analyse des données et visualisation

Un nœud audio `AnalyserNode` n'effectue aucune modification sur le flux audio et permet d'obtenir des données de fréquence sur ce flux. Il est obtenu par la méthode `ctx.createAnalyser()`.

- Propriétés `JavaScript`:
  - `fftSize`: [R/W] entier positif qui représente la taille de la `FFT` (Fast Fourier transformation) qui sera utilisé pour déterminer le domaine fréquentiel.
  - `frequencyBinCount`: [R] entier non signé égal à la moitié `fftSize`. C'est en général le nombre de valeurs qu'on manipule pour la visualisation.
- Méthodes `JavaScript`:
  - `getByteFrequencyData(arr)`: copie les données de fréquence dans le tableau d'octets non signés (positifs) `arr` de type `Uint8Array`.
  - `getByteDomainData(arr)`: copie les données de forme d'onde ou du domaine temporel dans le tableau d'octets non signés `arr` de type `Uint8Array`.

`GainNode`: amplitude



Un nœud audio `GainNode` amplifie chaque trame d'échantillons du signal audio. Il est obtenu par la méthode `ctx.createGain()`.

- Propriétés `JavaScript`:
  - `gain` [R/W]: amplification à appliquer (nombre)

`BiquadFilterNode`: filtres prédéfinis

Un nœud audio `BiquadFilterNode` applique un `type` de filtre parmi une liste prédéfinie de filtres sur un signal en entrée et retourne une sortie.

- Propriétés `JavaScript`:
  - `type`: nom du filtre à appliquer (chaîne de caractère)
- Liste de filtres:
  - https://developer.mozilla.org/en-US/docs/Web/API/BiquadFilterNode

`ConvolverNode`: produit de convolution

Un nœud audio `ConvolverNode` applique un produit de convolution sur le signal en entrée et le retourne en sortie. Il utilisé généralement pour effectuer des effets de réverbération.

`WaveShaperNode`: interpolation linéaire

Une nœud audio `WaveShaperNode` renvoie en sortie une interpolation linéaire du flux audio donnée en entrée. Il est obtenu par la méthode `ctx.createWaveShaper()`.

- Cette distorsion est paramétrée par les propriétés `JavaScript` suivantes:
  - `curve(arr)`: description de l'interpolation linéaire par un tableau de flottants `arr` de type `Float32Array`. L'algorithme d'interpolation est celui donné sur cette page.

Sujet

Vous pouvez utiliser la ressource audio suivante si vous avez un problème: http://jplayer.org/audio/mp3/RioMez-01-Sleep_together.mp3

- Créez un document `HTML` contenant un élément `<canvas>` de dimension `252*256` ainsi qu'un élément `<audio>` référençant une ressource audio de votre choix. Cet élément `<audio>` devra permettre de contrôler l'état de la lecture (pause, volume et positionnement).
- Stylisez le cadre de façon à afficher ses bordures.
- Dans votre fichier de script, utilisez l'API `Web Audio` pour initialiser le contexte audio. Écrivez ensuite une fonction `setupNodes()` qui se chargera de créer l'ensemble des nœuds et liaisons nécessaires pour relier la source audio à la destination audio.
- Appelez `setupNodes()` une **unique** fois lors de la première lecture de l'élément audio. Rappel: les liaisons sont bien formées si vous pouvez entendre l'audio.
- Ajoutez un nœud d'analyse audio au graphe audio dans `setupNodes()`.
- Créez une fonction `visualizeSinewave()` qui permet dans le `<canvas>` d'afficher le signal sinusoïdal du flux audio. Vous l'appellerez à intervalle régulier.
- Créez une fonction `visualizeFrequencyBar()` qui permet dans le `<canvas>` d'afficher les données de fréquences sous forme de diagramme en bâton. Vous l'appellerez à intervalle régulier.
- Permettre à l'utilisateur, via une sélection, de choisir quelle visualisation il souhaite afficher dans le `<canvas>`.
- Permettre à l'utilisateur d'appliquer un filtre audio de son choix (implémentez en deux).
  - modification du gain
  - effet de réverbération
  - accentuation des basses
  - effet distorsion
- (OPTIONNEL) Changez les couleurs dynamiquement des parties du signal sinusoïdal et des bâtons du diagramme en fonction de leur valeur.
- (OPTIONNEL) Rajoutez d'autres filtres d'effets à sélectionner.