HTML5 : Audio et Vidéo

- Audio et vidéo sur le Web
- Audio et vidéo avec HTML5
- Audio et vidéo : quels formats ? Quels codecs ? Quels conteneurs ? Que faire en pratique ?
- Lecteurs audio et vidéo
- Les balises <audio> et <video>
- Les autres balises et attributs
- Pour aller plus loin

Audio et vidéo sur le Web

- Il a fallu attendre le **haut débit** et un accès à **Internet pour tous** pour rendre possible l'usage de l'audio et de la vidéo
- Au début, on installait des **plug-ins** dans le navigateur pour supporter des **formats propriétaires** et installer le bon **codec**
- En HTML, on employait les balises **<object>** et **<embed>** et la technologie privilégiée était **Flash** jusqu'à son abandon par Apple et Adobe
- HTML5 a apporté de nouvelles solutions

Audio et vidéo avec HTML5

- Ajout des éléments <audio> et <video> pour une lecture native dans le navigateur sans plug-in
- Pouvoir les manipuler avec CSS et JavaScript
- Mais pas d'accord sur une norme unique donc, variété de formats, de conteneurs et de codec
  - Codec : Compression-DECompression du signal
    ex : divX, H.264, MPEG4 AVC, VC-1, FLAC,...
  - Conteneur : structure de fichier qui regroupe les flux audio et vidéo, les sous-titres,...
    ex : AVI, MKV, MP4, MPG, OGG, WEBM,...

Vidéo pour le Web

- Actuellement, les leaders du marché sont :
  - **Theora** dans un conteneur Ogg, venant de l'Open Source avec Mozilla et implémenté dans Opera et Chrome ainsi que Edge
  - **WebM**, venant de Google, soutenu par Opera et Mozilla
  - **H.264** ou MPEG4/AVC, mis en avant par Apple pour Safari, iOS. Il est maintenant implémenté dans la plupart des navigateurs
- Mais ça peut changer au fil des années

  https://caniuse.com/#feat=video

Audio pour le Web

- Actuellement, les leaders du marché sont :
  - **MP3** (MPEG-1 Audio Layer 3) est supporté par la plupart des navigateurs
  - **AAC** (Advanced Audio Coding) choisi par Apple pour iTunes, est censé meilleur que MP3 à débit équivalent
  - **Vorbis** (avec conteneur Ogg) est libre et est supporté sur Chrome, Firefox et Opera
- Mais ça peut changer au fil des années

  https://caniuse.com/#feat=audio

En pratique

- Dans les balises <audio> et <video>, on peut renseigner **plusieurs fichiers** de formats différents avec la balise **<source>**
- On peut utiliser des logiciels ou des sites web de **conversion de formats**
  - HandBrake, MiroVideoConverter, FFmpeg, VLC, FreeMediaConverter, OggConvert, Audacity,...
  - https://www.online-convert.com/fr
  - https://www.onlinevideoconverter.com/fr
  - https://cloudconvert.com/

Lecteurs audio et vidéo

- Les lecteurs varient en apparence selon le navigateur car il n'y a pas de norme
- Si le navigateur (ancienne version) ne supporte pas l'audio ou la vidéo, il est possible d'ajouter un texte alternatif ou un lien de téléchargement entre les balises
- Pour avoir un plus grand contrôle, il faudra utiliser CSS et JavaScript
- Voir aussi :
  https://developer.mozilla.org/fr/docs/Web/HTML/Formats_pour_audio_video

Elément audio

- La balise **<audio>** contient l'attribut **src** pour faire référence au fichier son
- La balise **<source>** permet de définir plusieurs formats audio avec plusieurs types **MIME**

```
<audio src="musique.mp3">
    <a href="musique.mp3">Télécharger le son</a>
</audio>

<audio>
    <source src="musique.mp3" type="audio/mpeg">
    <source src="musique.aac" type="audio/mp4">
    <source src="musique.oga" type="audio/ogg">
</audio>
```

Elément video

- La balise **<video>** contient aussi l'attribut **src,** ainsi que les dimensions **width** et **height**
- La balise **<source>** permet de définir plusieurs formats vidéo avec plusieurs types **MIME**

```
<video src="film.ogv" width="640" height="480">
    <a href="film.ogv">Télécharger la vidéo</a>
</video>

<video>
    <source src="video.mp4" type="video/mp4">
    <source src="video.ogv" type="video/ogg">
    <source src="video.webm" type="video/webm">
</video>
```

Attributs

- src : définit le fichier média à utiliser
- width / height : définit les dimensions de la vidéo
- controls : affiche les contrôles visuels
- poster : affiche une image statique pour la vidéo
- autoplay : démarre la lecture automatiquement
- preload : charge le média à l'avance
- loop : joue le média en boucle

Pour aller plus loin

- Il faudra utiliser les interfaces de programmation (API) et les piloter avec du code JavaScript si on souhaite :
  - manipuler l'interface de contrôle (lecture, pause, stop, avance ou retour rapide,...)
  - gérer les événements (changement d'état, volume, durée,...)
  - créer une interface graphique personnalisée
  - détecter les erreurs